Palestine Polytechnic University
Deanship of Graduate Studies and Scientific Research
Master of Informatics

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

Submitted by:

Karmel Fathi Shehadeh

Supervised by:

Prof. Nabil Arman

Dr. Faisal Khamayseh

Thesis submitted in partial fulfillment of requirements of the degree of Master of Science in Informatics
August, 2022

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 Arabic User Requirements into Functional and Non-Functional Requirements using NLP Tools**, submitted by **Karmel F. Shehadeh** in partial fulfillment of the requirements for the degree of Master in Informatics.

Graduate Advisory Committee:

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

Signature:_____ Date:_____

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

Signature:_____ Date:_____

Dr. Mahmoud Al-Saheb (Internal committee member), Palestine Polytechnic University.

Signature:_____ Date:_____

Dr.Rashid Jayousi (External committee member)

Signature:_____ Date:_____

Thesis Approved

Dr. Nafez Naser Aldeen
Dean of Graduate Studies and Scientific Research
Palestine Polytechnic University

Signature:_____ Date:_____

DECLARATION

I declare that the Master Thesis entitled **"Semi-Automated Classification of Arabic User Requirements into Functional and Non-Functional Requirements using NLP Tools"** is my original work, and hereby 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.

Karmel F. Shehadeh

Signature:_____ Date:_____

STATEMENT OF

PERMISSION TO USE

In presenting this thesis in partial fulfillment 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.

Karmel F. Shehadeh

Signature:_____ Date:_____

الملخص

تلعب هندسة المتطلبات دوراً مهماً للغاية في دورة حياة تطوير البرمجيّات، حيث يعتمد نجاح أو فشل مشاريع البرمجيّات بشكلٍ بارزٍ على مرحلة هندسة المتطلبات. غالباً ما تحتوي مستندات المتطلبات على نوعين من المتطلبات، أحداهما هي المتطلبات الوظيفيّة، والتي تحدد ميزات النظام، والأخرى هي المتطلبات غير الوظيفيّة، والتي تحدد سمات الجودة لميزات النظام وبيئة التطوير. يتمّ توثيقهما في الغالب بلغة طبيعية. عادةً ما يتطلب التصنيف اليدويّ للمتطلبات الكثير من الجهد البشري، وهي مَهمَّة صعبة وحسّاسة. تم تحسين عملية تصنيف المتطلبات في السنوات الأخيرة من خلال التصنيف باستخدام الأساليب الآلية أو شبه الآلية والتي استخدمت لنفس الغرض من هندسة البرمجيّات الآلية حيث تساعد المطوّرين على تقديم برامج عالية الجودة تلبيّ توقّعات المستخدمين بأقل وقت وتكلفة.

في هذه الرسالة، نقدم نهج تصنيف شبه آلي جديد للمتطلبات الوظيفيّة وغير الوظيفيّة باللّغة العربيّة باستخدام أدوات معالجة اللغة الطبيعية المسمى أدوات CAMeL. اقترحنا مجموعة من القواعد تستند إلى التركيبات الأساسية للجمل العربيّة من أجل استخراج المعلومات من متطلبات البرمجيّات المكتوبة باللغة العربيّة لتصنيف المتطلبات إلى متطلبات وظيفيّة وغير وظيفيّة. تُستخدم أدوات CAMeL لإنشاء الرموز المميزة وعلامات PoS و lemmas لمتطلبات المستخدم التي تم تحليلها، ثمّ تطبق مجموعة من القواعد الإرشادية المقترحة على مخرجات CAMeL لتحديد أقرب

فئة لكل عبارة. لقد قمنا بتنفيذ النهج المقترح بكتابة نص برمجيّ بلغة python ، وأدوات CAMeL 1.3.1 ، ونظام تشغيل Ubuntu 20.04 LTS .

تمّ التحقق من صحة النهج المقترح باستخدام مجموعة من التجارب التي تتضمن مجموعة من النماذج الحقيقية، تمّ تقييمها من قبل مجموعة من خبراء هندسة البرمجيّات ومجموعة من طلبة الدراسات العليا وطلاب البكالوريوس الذين هم على دراية بمتطلبات البرمجيّات. أظهرت النتائج أن المنهج المقترح يحقق نتائج أفضل في تصنيف متطلبات البرمجيّات المكتوبة باللّغة العربيّة مقارنة بطلاب الدراسات العليا وطلاب البكالوريوس.

Abstract

Requirement engineering plays a very important role in the software development life cycle. The success or failure of a software project depends prominently on the requirement engineering phase. Requirement documents commonly have two types of requirements, one is Functional Requirements, which defines the features of the system-to-be, and the other is Non-Functional Requirements, which defines the quality attributes of the system features and development environment. They are predominantly documented in natural language. A lot of human effort is required for manual classification, which is a challenging and delicate task. Software requirements classification process has been improved in recent years by classification requirements using automated or semi-automated methods for the same purpose of Automated Software Engineering which helps developers to deliver quality software that meets users' expectations completely with saving time and cost.

In this thesis, we presented a new Semi-Automated classification approach of Arabic functional and non-functional requirements using a natural language processing tools, namely CAMeL Tools. We proposed a set of heuristics based on basic constructs of Arabic sentences in order to extract information from software requirements written in Arabic to classify the requirements

into functional and non-functional requirements. CAMeL tools are used to generate tokens, PoS tags, and lemmas of the parsed user requirements, then a set of proposed heuristic rules are applied to CAMeL outputs to determine the closest class for each statement. We implemented the proposed approach using Python code using CAMeL Tools 1.3.1, under Ubuntu 20.04 LTS.

The proposed approach is validated using a set of experiments involving a set of real cases evaluated by a group of software engineering experts, graduate, and undergraduate students who are familiar with software requirements. The results showed that the proposed approach achieves better results in the classification of Arabic software requirements than graduate, and undergraduate students.

DEDICATION

To my family,

I could never have done this without your faith, support, and constant encouragement.

ACKNOWLEDGEMENT

Many people have directly or indirectly contributed to the successful completion of this thesis. They will all be remembered in my heart. First, I would like to take this opportunity to highly appreciate my thesis supervisors Prof. Nabil Arman and Dr. Faisal Khamayseh, for their support, motivation, patience, useful comments, and immense knowledge.

Besides, I would like to thank, my former colleague in the master's program Eng. Ibrahim Nassar, my teachers: Eng. Ezdehar Jawabreh, Dr. Mahmoud Al-Saheb, and Dr. Ghassan Shahin for all their help, support, and useful notes.

I would like to express my deep sense of reverence and gratitude to all of my respected teachers for the invaluable knowledge they imparted to me.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interfaces
ASE	Automated Software Engineering
Acc	Accuracy
CF	Certainty Factor
CLIs	Command-Line Interfaces
FN	False Negative
FP	False Positive
FR	Functional requirements
NFR	Non-functional requirements
NLP	Natural Language Processing
P	Precision
PoS	Part of Speech
R	Recall
RE	Requirements Engineering
SDLC	Software Development Life Cycle
SRS	Software Requirements Specification
TN	True Negative
TP	True Positive

Chapter 1

Introduction

Automated Requirements Classification is one of the most important research problems in software engineering. Software engineering has shown to be a suitable and effective field for automation whereas several approaches and techniques specific to the Automated Software Engineering (ASE) area are increasingly being implemented into various processes in the software life cycle [1]. Particularly, the Requirements Engineering (RE) subdisciplines activities (elicitation, analysis, specification, and validation) [2] have become the focus of several tools and frameworks aimed at assisting them automatically [1]. ASE is used to automate the software systems activities to improve the quality and productivity of the software. Also, it lowers the cost and time of analysis of user requirements [3].

RE is split into requirements development and requirements management. Requirements development subdivide into elicitation, analysis, specification, and validation. These subdisciplines encompass all the activities focused on understanding the customer's needs regarding the software to be developed. While requirements management activities include: defining the requirements baseline, evaluating the impact of proposed requirements changes,

and keeping project plan current with the requirements as they evolve [2].

In the requirement elicitation, customers' specifications and needs are well understood. In the analysis task, those needs are checked and redefined. Then at the requirement specification task, all user specification requirements are documents in a clear and correct form. Finally, schedules and prioritizes the requirements on requirement management tasks. After the requirement engineering phase, customer specifications are written in a Software Requirements Specification (SRS) document [4]. However, this activity is usually time-consuming and error-prone, so using an automated approach helps speed up the process of building systems and can reduce the number of possible errors.

System requirements are divided into functional requirements (FR) and Non-Functional Requirements (NFR). FRs of a system describe what the system should do, and NFRs show how the system behaves with respect to some observable attributes like reliability, reusability, maintainability, etc. Both FR and NFR are organized and specified in a Software Requirements Specification (SRS) document [5].

There is a clear and unanimous definition of the FRs and NFRs. FRs are statements about what services the system should provide, how it should act in specific situations, and how it should respond to specific inputs. The system's FRs may also state what it should not do. On the other hand, NFRs are constraints and limitations on the system's services and functions. They include timing constraints, constraints imposed by standards, and constraints on the development process. NFR often applies to the whole system rather than particular system services or features. In a nutshell, FRs describe the functionality of a system, whereas NFRs describe the system's constraints

and properties [6].

Since there is no direct automation procedure from Arabic requirements to classification, we propose an approach to classify user requirements written in Arabic with the least interventions using an Arabic natural language processing tools namely CAMeL. This tool is used to parse different statements of the user requirements written in Arabic language to obtain the difference between FRs and NFs. A set of steps that describe our approach for classifying user requirements is presented in this thesis.

1.1 Motivation

Manual Software requirements classification is time-consuming and costly. In addition, it is an error-prone process, especially for large and complex requirements of systems. The main motivation is developing a semi-automated approach that can classify software requirements instead of the traditional/-manual methods, which saves time and cost. In addition, there is no research, to the best of our knowledge, about the classification of Arabic requirements into FR and NFR.

1.2 Problem Statement

FRs and NFRs are equally important in software engineering where both of them are organized in a Software Requirements Specification document using natural languages. Manual classification for FRs and NFRs is a very exhaustive and time-consuming task for software engineers. Usually, they face problems in the analysis of user requirements phase. In addition, any error in the classification process may lead to misunderstanding or ambiguity

in the requirements by the software developers.

There are several techniques for software requirements classification. According to the review of the literature, one of the techniques for software requirements classification is using machine learning. However, the learning approach needs to train the model. If the training data are not available, then the researchers have to prepare training data manually. This drawback of machine learning methods in software requirements classification is related to the amount of pre-categorized requirements required to achieve good levels of precision in the classification process.

In our research, we attempt to solve this drawback by providing a Semi-Automated classification of software requirements using an NLP tool. In this thesis, we will classify requirements written in Arabic language. However, Arabic language is not an easy task to parse because of the following reasons: First, the particularities of the Arabic language make it more ambiguous than other natural languages. This is due to its morphological, syntactic, and semantic characteristics. Second, is the significant lack of digital resources in the Arabic language, especially concerning the grammar and corpora [7]. The main problem is how to find the approach that can automatically classify Arabic user requirements into FR and NFR using NLP tools.

The main objective of this research is to propose a novel approach to classifying FRs and NFRs written in Arabic using a semi-automated method based on NLP tools in order to make Arabic requirement classification tasks perform faster, easier, and more accurately.

1.3 Proposed Solution

We propose a high-level approach to illustrate the detailed steps of classifying Arabic user requirements using CAMeL tools. We also developed a system for classifying Arabic user requirements using python language [8].

1.4 Research Steps

We followed the following steps for our research:

1. Review previous researches on software requirements classification and automated software engineering methods.
2. Review the Arabic user requirements and how they must be written.
3. Review the difference between FR and NFR.
4. Review the Arabic Natural Language Processing tools and how to use the appropriate Arabic NLP tool.
5. Study students and developers' SRSs to extract the set of heuristics that distinguish FRs from NFRs.
6. Propose an approach to classify Arabic user requirements.
7. Request from software engineering experts to evaluate our approach.
8. Implement the proposed approach using Python and CAMeL Tools.
9. Finally, evaluate our approach.

1.5 Research Objective

The main objective of this thesis is to develop a novel approach to the classification of user requirements written in Arabic into FRs and NFRs using a natural language processing tool to analyze the sentences of the Arabic user requirements.

1.6 Contributions

In this thesis, we proposed a novel approach for the classification of Arabic user requirements into FRs and NFRs. In this approach, a natural language processing tool is used to analyze the sentences of the Arabic user requirements. Based on the outcome of the analysis, a set of heuristics are presented to guide the classification process. These heuristics use the tokens produced by the chosen NLP tool. We also developed a system for classifying Arabic user requirements using Python language. This research aims to help software engineers in the analysis phase to reduce the cost and the time required in performing manual classification.

1.7 Research Importance

The benefits of categorizing requirements include helping in discovering commonalities and unexpected relationships between requirements, improving the traceability of the requirements document, and it may help in finding missing requirements [6].

Several attempts have worked on finding an approach to classify the requirements written in English, but there is no one that has worked on classifying the requirements written in Arabic due to the specificity of the Arabic language and the complexity of its grammar. This research aims to help software engineers in the analysis phase to reduce the cost and the time required in performing these manual processes and activities. Since there is a lack of research serving Arabic requirements, we are towards working on taking requirements written in the Arabic language as input and classifying them.

1.8 Thesis Organization

The remaining parts of the thesis are organized as follows: Chapter 2 contains a summary of some previous works related to our thesis. Chapter 3 describes the system's theoretical background related to the main concepts that are needed to understand the rest of the thesis. Chapter 4 covers the methodology used in this thesis. Chapter 5 demonstrates the experiment, evaluation, and results achieved by the work and the discussion of the results. Chapter 6 concludes the work and proposes some new directions for future work.

Chapter 2

Background

This chapter gives a theoretical background needed for understanding the rest of the thesis. The first section explains the requirements engineering and how to write user requirements in Arabic. The second section talks about natural language processing tools and focuses on CAMel tools.

2.1 Requirements Engineering (RE)

RE is the most important area of the software engineering phase and the whole software development life cycle (SDLC). RE phase is used to translate the inaccurate, incomplete needs of the software users into formal complete, and precise specifications. The specifications are considered as a contract between the software users and the developers. Therefore, the importance of RE is huge at the early stage of the development of software in developing effective software and in minimizing software problems at the early stage of the development of software [9].

RE is split into requirements management and requirements development. The requirements management activities include: defining the requirements

baseline, evaluating the impact of proposed requirements changes, and keeping project plans current with the requirements as they evolve. Requirements development subdivided into [2]:

- Elicitation includes all of the activities involved with discovering requirements, like interviews, document analysis, workshops, prototyping, and others.
- Analyzing requirements encompasses achieving a richer and more precise understanding of each requirement also representing requirements in multiple ways.
- Requirements specification includes representing and storing the requirements knowledge in a well-organized fashion.
- Requirements validation confirms that you have the correct set of requirements information that helps developers to build software that satisfies the business objectives.

2.1.1 Software Requirements Specification (SRS)

SRS also known as the software requirements document, is a formal description of what system developers should implement. It includes both the user and system requirements. Both of them can be included in a single description. In some cases, the user requirements are defined in the introduction of the system requirements specification. Also, if there are a large number of requirements, the detailed system requirements may be presented in a separate document [6].

The level of details in the requirements document is determined by the type of system and the development process employed. A detailed requirement document is needed when the system is critical because safety and security

need accurate analysis and when a separate company will develop the system. But, if the development process is done within the same organization, the system specifications can be much less detailed and any ambiguities can be resolved during the development process [6].

2.1.2 Types of Requirements

There are two types of requirements: user requirements and system requirements [10]:

- User Requirements:

  User requirements are statements written in natural language and diagrams. They specify the expected system services and system constraints. User requirements readers don't concern about how the system will be implemented and the detailed facilities of the system.

- System Requirements:

  System requirements are detailed descriptions of the services and constraints of the system. They are derived from analysis of user requirements and they should be structured and precise. System requirements readers need to know more about what the system will do because they are participatory in the system implementation or they are concerned with how system requirements will support the business processes.

Another taxonomy for requirements focuses on the type of requirement. It categorized them into FRs and NFRs [6]:

- Functional Requirements: These are statements of what the system should do, what services the system should provide, how it should act in specific situations, and how it should respond to specific inputs. FRs may also state what it should not do.

- Non-Functional Requirements: These are statements of constraints and limitations on the system's services and functions. They include timing constraints, constraints imposed by standards, and constraints on the development process. Also, it shows how the system behaves with respect to some observable attributes like reliability, reusability, maintainability, etc. NFR often applies to the whole system rather than particular system services or features.

  NFRs may be derived from the desired characteristics of the software (product requirements), the organization that developing the software (organizational requirements), or from external sources.

2.1.3 Benefits of Good User Requirements

Natural language has been used to write software requirements since the beginning of software engineering. Because it is intuitive, expressive, and universal. It also could be ambiguous, and vague and its meaning depends on the background of the reader [6].

Good requirements provide many benefits. These benefits affect development, productivity, testing, quality, and the organization. These benefits help developers during the development of new systems and during the maintenance of existing systems. User requirements written in Arabic must be written in a good way based on the grammar structure in Arabic language to avoid problems and errors in the analysis phase.

The IEEE standard for SRS identifies a good requirement as complete, consistent, correct, verifiable, unambiguous, and traceable [11]. Completeness means the requirement is complete and does not need further amplification and it will provide sufficient capability. Constancy means the requirement

does not contradict other requirements. There are no duplicate requirements [12]. Each requirement must be correct, it is correct if it accurately states a function that the system must provide [11]. Verifiability means the requirement is not vague or general but is quantified in a manner that can be verified. That if we can test the requirements, testers will base their test on the requirements set as well. Unambiguous means each requirement must have only one interpretation. The requirements statement must not leave doubt in the reader's mind. Traceability helps the developers and testers to maintain the system. At any time we require changing or removing a requirement, we will be able quickly to determine the parts of the design and implementation that support the requirement [12].

2.1.4 User Requirements Written in Arabic

Arabic language is a prominent member of the Semitic language family. It consists of 28 letters and it is written from right to left. Grammar in Arabic language is a collection of rules that describes informed sentences well.

User and system requirements are usually written in natural languages supplemented by relevant diagrams and tables. Effective and correct written format of Arabic user requirements is a critical skill because of Arabic complex linguistic structure. In this section, we present the general guidelines for writing Arabic user requirements. A set of guidelines has been defined for writing Arabic user requirements to reduce the ambiguity of Arabic sentences and prevent errors in later analysis phases, to increase the probability of having correct and good statements that can quickly and easily be processed.

The following general guidelines should be followed during the writing of

Arabic user requirements [13]:

1. Write in modern standard Arabic.
2. Use English technical words within Arabic sentences.
3. Write short sentences as much as possible.
4. Write complete sentences with correct grammar and spelling.
5. Write only active-voice sentences.
6. Write sentences of form subject-verb-object or verb-subject-object only.
7. Write sentences in a consistent fashion.
8. Sentences should have clear boundaries.
9. Avoid abbreviations unless defined in the glossary.
10. Avoid using pronouns, possessive pronouns, and possessive adjectives.
11. You should put a full stop at the end of each sentence.

2.2 Natural Language Processing Tools

There are many tools for Arabic morphological analysis. Each has different characteristics. This is based on how these tools are developed and what database is used. There are several tools that are freely available and very suitable for tokenizing Arabic text:

- Stanford CoreNLP is a multilingual Java library, CLI, and server providing multiple NLP components with varying support for different languages. Arabic support is provided for parsing, tokenization, PoS tagging, and sentence splitting [14].
- MADA+TOKAN is a versatile and freely available system that can derive extensive morphological and contextual information from raw Arabic text, and then use this information for a multitude of crucial

NLP tasks. Applications include high-accuracy part-of-speech tagging, discretization, lemmatization, disambiguation, stemming, and glossing. MADA operates by examining a list of all possible analyses for each word and then selecting the analysis that matches the current context best by means of support vector machine models classification for 19 distinct, weighted morphological features. All disambiguation decisions are made in one step because the selected analysis contains complete diacritic, lexemic, glossary, and morphological information. TOKEN takes the information provided by MADA to generate tokenized output in a wide variety of customizable formats [15].

- MADAMIRA Tools are a system for morphological analysis and disambiguation of Arabic. It enabled features such as part-of-speech tagging, segmentation, lemmatization, tokenization, NER, and base-phrase chunking. It supports both MSA and Egyptian and primarily provides a CLI, a server mode, and a Java API [16].

- CAMeL Tools are a Python-based collection of open-source tools for Arabic natural language processing. These tools currently provide utilities for pre-processing, morphological modeling, dialect identification, named entity recognition and sentiment analysis [17].

2.2.1 CAMeL Tools

CAMeL Tools is one of New York University Abu Dhabi's inventions. Nizar Habash, Ossama Obeid, Nasser Zalmout, and others developed this tool. It has many versions, the first one, V0.4.dev2 released on 12 Sep 2019, and the last one, V1.3.1 was released on 31 April 2022 [18].

CAMeL Tools is a collection of open-source tools for Arabic natural lan-

guage processing in Python. They chose Python due to its ease of use and its pervasiveness in NLP and Machine Learning along with libraries. CAMeL tools provides utilities for pre-processing, dialect identification, morphological modeling, named entity recognition, and sentiment analysis [17].

CAMeL Tools is implemented in Python 3. It could be installed on Linux, macOS, and Windows. They provide Command-Line Interfaces (CLIs) and Application Programming Interfaces (APIs).

CAMeL Tools provides many utilities for Arabic NLP tasks, such as the following:

1. Preprocessing

   CAMeL Tools provides a set of preprocessing utilities for preparing and cleaning Arabic text. Before passing text to other CAMeL Tools components, some of these preprocessing utilities may need to be used.

   1.1 Simple Transliteration:

   Simple transliteration is the process of translating each Arabic character to/from a single non-Arabic character encoding.

   1.2 Unicode Normalization:

   In the Arabic language there ate variants and composed forms usually used for display purposes but they are problematic in text processing tasks. These variants and composed forms are generally used for display purposes and are problematic in text processing tasks.

   1.3 Orthographic Normalization

   When typing Arabic text, Arabic speakers frequently use shortcuts. For example, the different variants of the letter alef (’إ’, ’أ’,’آ’,’ا’) may be typed as just ’ا’. Some of these substitutions can

be the result of typos. So, CAMeL tools provide orthographic normalization which is the process of converting letter variants or visually similar letters into a single form.

1.4 Dediacritization

Dediacritization is the process of eliminating Arabic diacritical marks. Arabic diacritical marks. Diacritics usually increase data sparsity and so most Arabic NLP techniques ignore them.

1.5 Word Tokenization

CAMel tools provide the utility function to split sentences by whitespace and separate punctuation, whereas some CAMeL Tools components expect input text to be pretokenized by whitespace and punctuation.

2. Morphology

CAMeL Tools provides an effective morphological analysis, reinflection, and generation system.

2.1 Analysis

Morphological analysis is the process of generating all possible readings of a given word out of context. Each of these analyses is defined by a set of morphological and lexical features.

2.2 Generation

Generation is the process of inflecting a lemma for a set of morphological features.

2.3 Reinflection

Reinflection is the process of converting the input word in any form to a different form like tense, gender, etc. Rreinflector works similarly to the generator except that the word doesn't have to be

a lemma and it does not have to be restricted to a specific 'PoS'.

3. Disambiguation

   Disambiguation is the process of determining what is the most likely analysis of a word in a given context. In CAMeL Tools, disambiguation is the backbone for many Arabic NLP tasks such as PoS tagging, diacritization, and morphological tokenization.

4. Tagging

   CAMel tools provide tagging utilities to generate PoS tags for all words in a given sentence. It uses a set of tags to describe the Arabic words in the statements. Each word has a tag, and every tag has a special meaning. Table 2.2 shows all tags for CAMel tools with examples.

5. Morphological Tokenization

   CAMel tools provide another type of tokenization than Word Tokenization which is morphological tokenization whereby Arabic words are split into component prefixes, stems, and suffixes.

6. Dialect Identification

   CAMel tools provide a dialect identification system that can distinguish between 25 city dialects as well as Modern Standard Arabic.

7. Sentiment Analysis

   CAMel tools provide a pre-trained sentiment analysis system that has been trained using a combination of multiple data sets. The output of this system is one of three sentiment tags for each sentence: 'positive', 'negative', and 'neutral'.

8. Named Entity Recognition

   CAMeL Tools comes with an easy-to-use named entity recognition

(NER) system. For each token in input sentences. NERecognizer outputs a label that indicates the type of named entity. For each token, the system outputs one of the following labels: 'B-LOC', 'B-ORG', 'B-PERS', 'B-MISC', 'I-LOC', 'I-ORG', 'I-PERS', 'I-MISC', 'O'. Named entities can either be a LOC (location), ORG (organization), PERS (person), or MISC (miscellaneous).

We've used CAMeL Tools because it's one of the best free NLP tools to date. The table 2.1 [17] below shows the extent of its superiority over the MADAMIRA tool. As for MADA, MADAMIRA, TOKAN, now is called MADAMIRA which supersedes MADA. They no longer distribute MADA, MADAMIRA is better and faster [19].

Table 2.1: Comparison of the Performance of CAMeL Tools Multitask Learning to MADAMIRA.

Utilities	MADAMIRA	CAMeL Tools Multitask
DIAC	87.7%	90.9%
LEX	96.4%	95.4%
PoS	97.1%	97.2%
FULL	85.6%	89.0%
ATB TOK	99.0%	99.4%

The systems are evaluated on their accuracy to correctly predict diacritics (DIAC), lemmas (LEX), Part of Speech (PoS), the full set of predicted features (FULL), and the ATB tokenization.

Table 2.2: PoS Tags of CAMeL Tools.

Arabic PoS	CAMEL PoS Tags	Examples
أداة تعريف	PART_DET	ال
حرف عطف	CONJ	و، أو
حرف جر	PREP	الى، من
أداة نفي	PART_NEG	لا، لن
أداة استقبال	PART_FUT	س، سوف
أداة ربط	CONJ_SUB	+و
ضمير اشارة	PRON_DEM	هذا
ضمير استفهام	PART_EMPHATIC	ماذا، كيف
أداة توكيد	PART_EMPHATIC	إن
جواب شرط	PART_RC	+ف
أداة نداء	PART_VOC	يا
اسم	NOUN	كتاب، قلم
اسم عدد	NOUN_NUM	عشرة، مئة
اسم علم	NOUN_PROP	هند، الأهرامات، روسيا
اسم كم	NOUN_QUANT	عُشر، بضع، ضعف
صفة	ADJ	كبير، سريع
صفة عدد	ADJ_NUM	الرابع، الأول
صفة مقارنة	ADJ_COMP	أجمل، أطول
فعل	VERB	قال، يرسل
ضمير	PRON	نحن
ضمير إشارة	PRON_DEM	هذا
ضمير استفهام	PRON_INTERROG	من
ضمير موصول	PRON_REL	ما
أداة استفهام	PART_INTERROG	هل، لماذا
أداة استثناء	PART_RESTRICT	الا
رقم	DIGIT	١٠٠، ٥٠٠
أجنبي	FORIEGN	Hello
علامة ترقيم	PUNC	، . %

Chapter 3

Literature Review

Manual classification for software requirements is a very exhaustive, time-consuming, and error-prone task for software engineers. Any error that occurs in the classification process may lead to misunderstanding or ambiguity in the requirements by the software developers. There are several methods for software requirements classification. One of the most important of these methods is to apply automated software engineering.

Automated software engineering methods have been applied in several areas in the software engineering field. In the last years, several efforts and research papers contributed to proposing methods for classification software requirements and generating UML models from user requirements. These methods can further be classified as machine learning approaches and rule-based approaches. Some researchers are interested in classifying non-functional requirements into different categories. Others are interested in classifying software requirements into functional and non-functional. This section presents some set of related research work in this area.

3.1 Software Requirements Classification Rule-Based Approaches

Singh. et al [20] combined automated software requirement identification and classification into NFR sub-classes with a rule-based classification technique based on thematic roles and determining the priority of extracted NFR based on their occurrence in multiple NFR classes.

As shown in Figure 3.1 [20], the proposed design consists of three phases: input SRS or a corpus of multiple documents to the first phase of the design for document pre-processing, thematic roles annotation using General Architecture for Text Learning (GATE) in the next phase, and finally, classification of annotated sentences into various NFR classes. They used PROMISE corpus for creating Java rules, testing these rules, and then prioritizing these extracted NFRs based on their occurrence in different NFR classes again using Java rules. They used Concordia RE corpus to verify that their classifier works on unstructured documents or documents other than SRS documents.

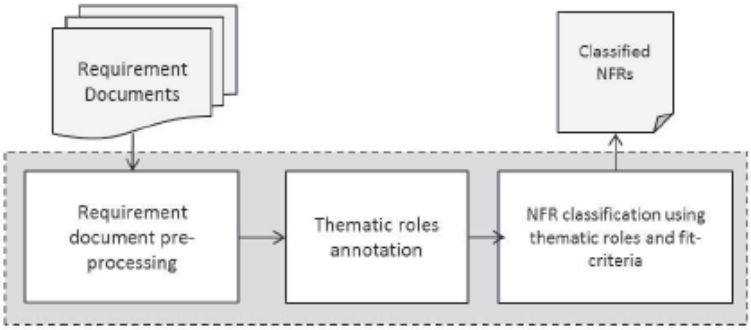

Figure 3.1: Three-phase system design.

Hussain et al [21] presented a methodology for automatic requirement classification using a text classifier with a part-of-speech (PoS) tagger. Researchers demonstrated that linguistic knowledge can assist in performing well in this classification task.

Some categories of words can be an indication to classify the sentence as NFR by their occurrences in the sentences. For example, NFR sentences often describe the quality attributes of the components or the system as a whole, and such sentences are likely to include adjectives and adverbs. Following these characteristics of NFR the authors chose a list of syntactic features as candidates and tested their probabilities of occurrence in the collection of NFR sentences, and thus, validated them to the most representative list of syntactic features. To classify the sentences, they developed a Java-based feature extraction program that parses the sentences from the corpora, extracts the values of all the features chosen by the authors, and uses Weka to train C4.5 decision tree learning algorithms.

Sharma et al [22] present a pattern-based rule approach to automatically parsing and classifying non-functional requirements based on NLP. They depend on identifying NFRs on extracting multiple features by analysis of natural language requirements where there is a certain combination of words and their relationship are unique for each category of NFR. These features are specialized as pattern-based rules that may be specialized in a human-readable language using a domain-specific language that they have defined. This work's contribution is confined to a small set of complex rules. In addition to the evaluation results where recall percentage was between 60 and 85% for five categories of NFR. They have been implemented their approach as a prototype tool.

Cleland-Huang et al [23] present an information retrieval approach to identify and classify Non-Functional Requirements automatically. This approach assumes that different categories of NFR are characterized by a set of distinct keywords 'indicator terms'. Those indicator terms are learned for a certain NFR category, they can be used to detect requirements, phrases, or

sentences, that are related to that category. The proposed approach includes three phases as the following: mining, classification, and application. Indicator terms are mined from pre-categorized NFR requirements during the training phase. Then detect and classify other requirements during the retrieval phase using the indicator terms. Finally, the classified requirements are used to support more advanced software engineering activities during the application phase.

3.2 Software Requirements Classification Machine Learning Approaches

Several studies worked on requirements classification based on the machine classification approach. We divide them into researches that classified the requirements written in English and others classified the requirements written in German.

3.2.1 Software Requirements Classification of English Specifications

Kurtanovi´c and Maalej [24] studied how accurately can automatically classify requirements as functional (FR) and non-functional (NFR) in the dataset with supervised machine learning. They used a second RE17 data challenge dataset. They also looked at how accurately they could identify different types of NFRs, such as usability, security, operational, and performance requirements. They developed and evaluated a supervised machine learning approach using meta-data, syntactical, and lexical features.

Haque et al [25] proposed an automatic NFR classification approach for qual-

ity software development by combining machine learning feature extraction and classification techniques. PROMISE software requirement dataset has been used. To find out the best pair of machine learning algorithms and selection approaches they applied an empirical study to automatically classify NFR with seven machine learning algorithms and four feature selection approaches.

The seven machine learning algorithms include MNB, GNB, BNB, KNN, SVM, SGD, and DTree. In addition to using Bow, TF-IDF (character level), TF-IDF (word level), and TF-IDF (n-gram) for feature extraction techniques which act as the input of machine learning algorithms. The whole process of this framework is divided into four steps as shown in Figure 3.2 [25] which include: Data Preprocessing, Feature Extraction, Train Classifier, and Classification requirements. As a result, this paper recommended TF-IDF (character level) for feature extraction with SGD SVM algorithm to predict the best results in NFR classification.

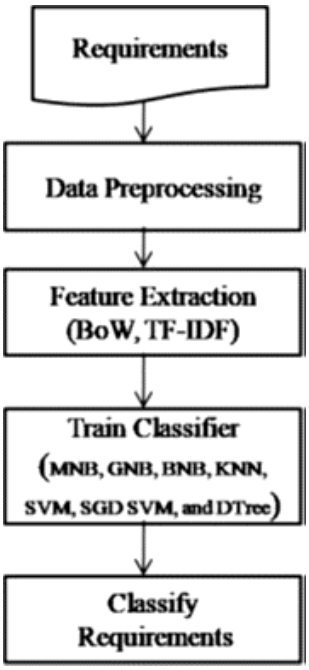

Figure 3.2: Proposed Method Overview.

Younas et al [26] proposed approach manipulates the textual semantic of functional requirements to identify the non-functional requirements. The semantic similarity is determined by the co-occurrence of patterns in large human knowledge repositories of Wikipedia. The similarity distance between popular indicator keywords and requirement statements is found in this study to determine the type of non-functional requirement. The proposed approach is applied to PROMISE NFR dataset.

In this paper, they used a semi-supervised machine learning method; there is no need for a training dataset. However, it can be supervised to some extent that the authors train their model with the Wikipedia dump of data. The proposed approach is described in the multistep procedure.

Abdur-Rahman et al [27] suggested deep learning approach using Recurrent Neural Network (RNN). This approach is performed in three steps: data pre-processing includes removing special characters, stop words, lemmatization, and tokenization. Step 2: word vectorization: they convert each word to vector using Word2Vec model. Then they trained three different classifiers: RNN, GRU, and LSTM models. They achieved a high precision rate equal 0.961, 0.967 recall, and 0.966 f1-score. In a conclusion, they found that RNN is a more effective approach to classifying NFR than CNN and GRU approaches.

3.2.2 Software Requirements Classification of German Specifications

Ott [28] evaluates several classifiers that classify requirements that written German into topics. Each topic is manually defined as a group of keywords. The best classifier is accustomed enable inspectors to reduce requirements'

defects in parallel. In this research, the author focused on the multinomial naive Bayes (MNB) and therefore the support vector machine (SVM) algorithms. The major problem that the research study points out is the difficulty of getting sufficient training examples so on improve the present recall of 0.8 and precision of 0.6.

The previous research enhanced by Knauss & Ott [29] they comparing three classification approaches: manual, semi-automatic, and automatic. The manual classification approach is where the user has to give one or several categories to every requirement. The semi-automatic method automatically classifies a given requirement but it requires modification or confirmation from the user. Finally, the automatic classifier does not require a manual confirmation or modification. Their research results show that a semi-automatic approach is the most promising, as provides the best ratio of quality and effort also the best learning performance.

Winkler et al [30] in this paper, the authors present an approach to automatically classify content elements of a natural language requirements specification as "requirement" or "information". This approach depends on convolutional neural networks. The authors used a set of 10,000 content elements extracted from 89 requirements specifications of our industry partner to train the neural network. In the evaluation of a real-world automotive requirements specification which writing in German, this approach was able to detect requirements with a precision of 0.73 and a recall of 0.89.

Table 3.1 illustrates the classification techniques used in each previous study that was reviewed. The first column shows the existing technique/s, and the second column shows research papers that used those techniques.

Table 3.1: Existing Techniques

Existing Technique/s	Research Papers
SVM Classifier	[24]
MNB, GNB, BNB, KNN, SVM, SGD, & DTree	[25]
Word2Vec	[26]
RNN, CNN & GRU	[27]
Rule-Based	[20] [29]
NLP Tool, DTree	[28] [30]
MNB, SVM	[21] [22]
CNN Classifier	[23]

3.3 Automated Generation of UML Diagrams

Several studies suggested approaches to automated generation of UML diagrams. We reviewed some research that generates UML diagrams from Arabic requirements and others from English User Requirements.

3.3.1 Automated Generation of UML Diagrams from Arabic User Requirements

Jabbarin and Arman [31][32] proposed a semi-automated approach to generating use case models from Arabic user requirements using natural language processing. A set of heuristics are presented to obtain use cases. These heuristics use the tokens produced by a natural language processing tool, namely Stanford parser.

Arman [33] proposed an approach to generate the use case diagrams by analyzing the Arabic user requirements. Using MADA+TOKAN for parsing different statements of the user requirements written in Arabic to obtain different components of sentences. A set of steps are presented to construct a use case model from Arabic user requirements.

Nassar and Khamayseh [34] proposed a semi-automated approach for con-

structing the activity diagrams from Arabic user requirements. They split and tokenize the Arabic user requirements using MADA+TOKAN parser. They present a set of heuristics based on basic constructs of Arabic sentences in order to extract information from Arabic user requirements to generate the activity diagrams. This study aims to assist software engineers in reducing the cost and time required to perform manual processes and activities during the analysis phase.

Alami et al [35][36] proposed a Semi-automated Approach for generating sequence diagrams from Arabic user requirements. They generated part of speech tags by parsing user Arabic requirements with a natural language processing tool. They proposed a set of heuristics for obtaining sequence diagram components such as objects, messages, and workflow transitions (messages). They generated sequence diagrams to be represented using XMI to be drawn using sequence diagram drawing tools.

3.3.2 Automated Generation of UML Diagrams from English User Requirements

Seresht and Ormandjieva [37] proposed an approach to generate automatically a high-level contextual view of the software system's actors and services (Context Use-Case Model - CUCM) from the textual user requirements. This approach depends on Recursive Object Model (ROM) and Expert Comparable Contextual (ECC) Models. This method can generate Context Use Case Model (CUCM) by applying the knowledge included in ECC model to specify actors and entailment rules for extracting CUCM elements.

Thakur and Gupta [38] proposed an Auto Sequence Diagram Generator (AutoSDG) for generating sequence diagram from Use Case Specification (UCS)

was written in natural language (English). The approach uses Stanford parser to generate POS tags and type dependencies (TDs). It then applies the proposed sentence structure rules and transformation rules on POS tags and TDs to identify elements to generate sequence diagram. The results show the quality of sequence diagrams obtained by AutoSDG is better than those obtained by existing automated approaches.

Gutiérrez et al. [39] proposed an automated method to auto-generate activity diagram of each use case written in a specific format. The generation is performed by model transformation, taking use case textual scenario as an input and producing the corresponding activity sub-diagram. The transformation is defined using QVT (Query/View/Transformation)-relational language and implemented in Java as a prototype tool.

Bajwa and Choudhary [40] proposed an approach for generating a class diagram from requirements written in English specifications. Software specifications are mapped to Semantic Business vocabulary and Rule (SBVR) which is adopted standard by OMG. Semantic Business Vocabulary and Rules are presented to generate controlled notations of software specifications. English requirements are parsed lexically, semantically, and syntactically and SBVR vocabulary is extracted. Then, the SBVR vocabulary is processed to construct an SBVR rule. In the last phase from the SBVR's rule, they extract the OO information (such as classes, attributes, methods, associations, etc). The approach is presented in a tool named NL2UMLviaSBVR.

Yue et al. [41] proposed an approach and a tool to automatically generate the sequence diagrams from use case specifications. A set of heuristic rules to identify the objects to presented UCSs. Also, a set of transformation rules is presented in the same study. They have implemented their proposed ap-

proach using a tool called a-Toucan. They validated their approach with six case studies. They compare their approach results with a diagram created by 4th-year undergraduate students and experts. The tool has better results than the ones manually created by students. Also, the results show that sequence diagrams generated by this tool are extremely consistent with the ones devised by experts.

Ilieva and Ormandjieva [42] presented a semi-automated approach to generate and create three requirements engineering models: use case model, hybrid activity diagram, and the domain model. Firstly, they use some of the available Part of Speech (POS) taggers to obtain the syntax category for each word in the text. The syntax category of each word is used to discover the role of the word in the sentence. Then arrange them in a table, there are three roles in a sentence: subject (Su), predicate (Pr), and object (Ob). A semantic network is created to represent the interconnections between the elements within the whole text. The last stage is interpretation of relationships extracted from the text in order to arrange this information and model it in the form of a diagram.

Chapter 4

Research Approach

Chapter 2 provides an overview of software requirements, expert systems, and NLP processing tools (CAMeL). This chapter focuses on the classification approach of the software requirements into functional and non-functional.

The new approach shows the steps of tokenizing and generating Posof the Arabic user requirements using CAMeL Tools. We proposed a set of heuristics based on basic constructs of Arabic sentences in order to classify Arabic user requirements into FR and NFR.

4.1 Arabic Sentence Syntax

Arabic sentences are mainly formed of two types: nominal and verbal sentences, the nominal sentence begins with a noun or pronoun, while the verbal sentence begins with a verb. In order for a sentence to be nominal sentence, it should have two main parts: the subject (مبتدأ) and the predicate (خبر). The verbal sentence should begin with a verb (فعل) and has a subject (فاعل). In more detail, the sentence also can be simple or complex, in which the simple sentence is the sentence where each part is formed of one word only and has

no inner sentences in its main elements. While the complex sentence is the sentence in which one of its main entities is another sentence [11].

4.2 Arabic User Requirements Classification Approach

Software user requirements are often classified as functional requirements or non-functional requirements where FR (Statements of services the system should provide), NFR (Constraints on the services or functions offered by the system) [5]. This section describes how the user requirements are automatically classified into functional and non-functional requirements based on Arabic sentence grammar and keywords. We reviewed several software graduations projects for PPU students and SRS documents for developers to extract features that distinguish functional from non-functional requirements. These features are used in proposing a set of heuristics for our approach. To analyze the Arabic sentence, we used CAMeL NLP tools for parsing, tokenization, part of speech, and sentence splitting. We extracted the following features for each non-functional and functional requirement and proposed heuristics for each one.

4.2.1 Non-Functional Requirement Linguistic Features

1. Non-functional requirements can contain numbers and digits, we represented it by H#1.
2. Non-functional requirements can contain non-Arabic words, we represented it by H#2.
3. Non-functional requirements can contain adjectives/adverbs, we represented it by H#3.

4. Non-functional requirements can contain NFR keywords, that have been mentioned in table 4.4 [43], and table 4.5 [23], we represented it by H#4.
5. Non-functional requirements sentence can be a verbal sentence (the main verb is verb-to-be) and the actor is the "system", we represented it by H#5.
6. Non-functional requirements can contain negation tools, we represented it by H#6.

4.2.2 Functional Requirement Linguistic Features

1. Functional requirements sentences have some common sentence structure, we represented it by H#7.
2. Verbs that are repeated in functional requirements sentences are usually verbs denoting action, we represented it by H#8.
3. Functional requirements sentences can be conditional sentences, we represented it by H#9.

In order to make our approach more accurate, we asked three software engineering experts to evaluate our heuristics through their experience. Then we've used the evaluation average as a Certainty Factor (CF) of our heuristics. We got evaluations mentions in Table 4.1:

We've found in the evaluation of H#1 and H#2 that the evaluation of expert#2 is lower than the evaluation of other experts, also in the evaluation of H#5 the first expert's evaluation is much lower than the evaluation of others. To confirm the validity of the heuristics and the accuracy of the experts' evaluation average. We've studied a larger number of examples of requirements and made more requirements classification experiments based on those heuristics. As a result, we found that the average for the experts'

Table 4.1: Expert Evaluations for Proposed Heuristics.

Heuristics#	Expert#1	Expert#2	Expert#3	Average
H#1	80%	60%	95%	78.33%
H#2	90%	60%	90%	80%
H#3	90%	70%	98%	86%
H#4	90%	90%	90%	90%
H#5	60%	90%	98%	82.66%
H#6	40%	50%	70%	53.33%
H#7.1	95%	80%	80%	88.33%
H#7.2	95%	80%	80%	88.33%
H#8	95%	70%	85%	83.33%
H#9	90%	70%	70%	76.66%

evaluations was very satisfactory for us.

Here, we manually selected a cutoff threshold (> 0.75). We have chosen this threshold because we found that all expert evaluations larger than 75% led to obtaining satisfactory classification results. So all the heuristics exceeding the cutoff threshold were selected as valid heuristics.

As we can see in the above table the average rate for all heuristics is higher than 75%, except for H#6, which was evaluated by experts as being less reliable than the other, so based on their opinions we have modified this heuristic to be more accurate and reliable.

Figure 4.1 describes the empirical methodology for our research. We developed a set of heuristics to classify the user requirements into functional and non-functional requirements by extracting user requirements features, using Arabic user grammar, analyzing PoS tags, collect FR and NFR keywords. The input of the approach is a set of unclassified user requirements written in Arabic language in which, the first step is normalise all requirements using CAMeL tools before passing text to other CAMeL Tools. The second step is generate tokens for all statements using CAMel tokens generator. The third step is generate PoS tags for all words in given sentences.

Then apply the proposed heuristics using generated PoS and Tokens. To get each sentence class compering FR_Score with NFR_Score then compare FR_CF with NFR_CF. The output of the approach is a set of classified user requirements.

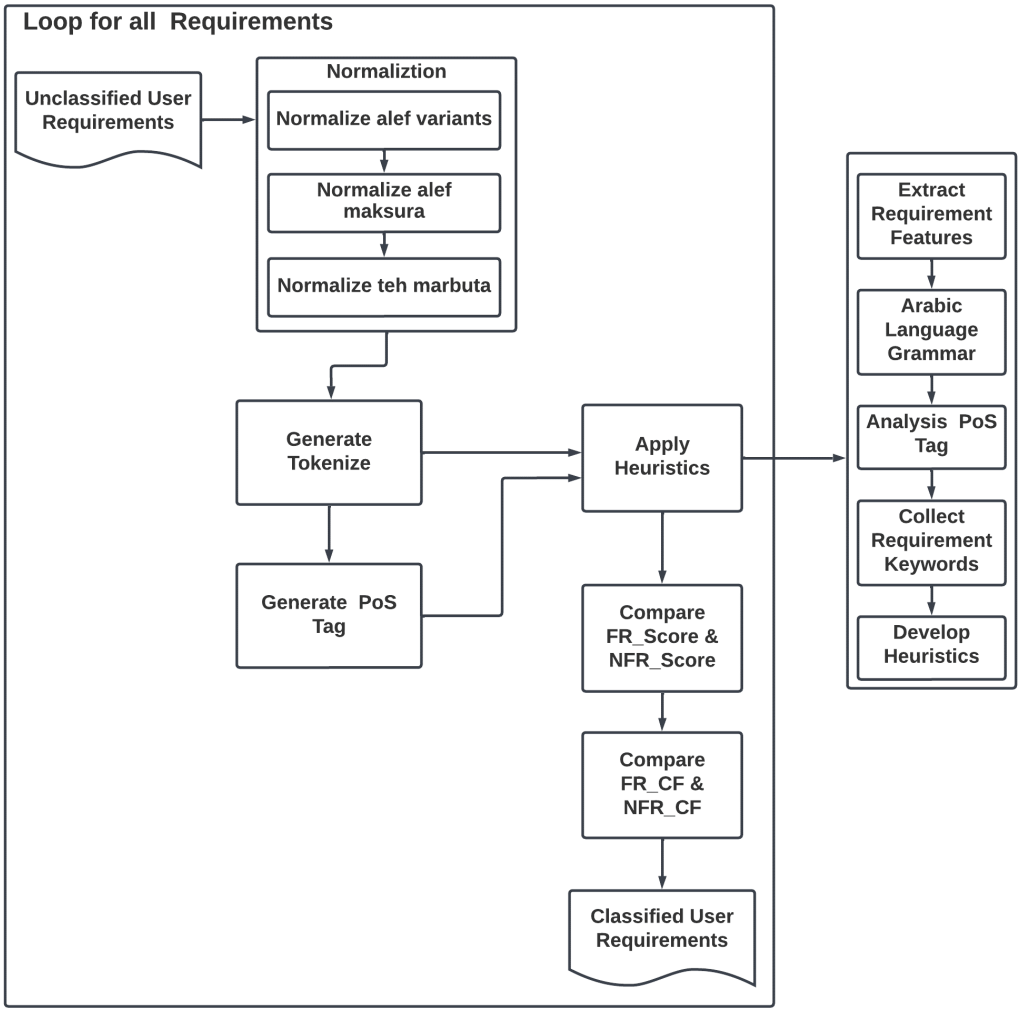

Figure 4.1: Empirical Methodology.

4.3 Proposed Heuristics

To classify the Arabic requirements into FR and NFR, we proposed a set of heuristics for Arabic sentences depending on the output of the CAMeL tools. These heuristics are presented as follows:

H1: If ['digit'] or ['noun_num'] tag exists at sentence PoS, then it is more likely to be NFR.

The appearance of cardinals/numbers is indicating a high probability of NFR. If cardinals/numbers are existing in sentences in any sentence syntax then it raises the likelihood that the sentence is a non-functional requirement whether it is written in numbers or letters. So, cardinals/numbers may distinguish non-functional requirements from functional based on their PoS tags. In order to check whether cardinals /numbers are present in the sentence, we have to look for all ['digit'] or ['noun_num'] tags and then add the certainty factor of this heuristic to the NFR_CF for the certain sentence.

If this condition applies to a sentence, the certainty factor of H1 is added to the NFR-CF of this sentence. NFR_CF = 78.33%

Some Possible structures for requirements that show the locations of cardinals/numbers in Arabic sentences:

1. 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)

A. Example: "يقوم النظام بتحديث العرض كل ثلاثين ثانية."

A translation of the example: "The system updates the display every thirty seconds."

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

CAMeL PoS: ['verb', 'noun', 'prep', 'noun', 'noun', 'noun_quant', 'noun_num', 'noun', 'punc']

B. Example:"يقوم النظام بتحديث العرض كل ٣٠ ثانية ."

A translation of the example: The system updates the display every 30 seconds.

CAMeL Tokens: ['يقوم', 'النظام', 'ب', 'تحديث', 'العرض', 'كل', '٣٠', 'ثانية', '.']

CAMeL PoS: ['verb', 'noun', 'prep', 'noun','noun', 'noun_quant', 'digit', 'noun', 'punc']

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

CAMeL PoS Tags:

(noun | pron) + Verb + (noun | prep + noun | noun_num + noun | digit + noun) + (noun | prep + noun | noun_num + noun | digit + noun) + (noun | prep + noun | noun_num + noun | digit + noun)

A. Example: "النظام يقوم بتحديث العرض كل ثلاثين ثانية ."

A translation of the example: "The system updates the display every thirty seconds."

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

CAMeL PoS: ['noun', 'verb', 'prep', 'noun', 'noun', 'noun_quant', 'noun_num', 'noun', 'punc']

B. Example:"النظام يقوم بتحديث العرض كل ٣٠ ثانية ."

A translation of the example: The system updates the display every 30 seconds.

CAMeL Tokens: ['النظام', 'يقوم','ب','تحديث','العرض','كل','٣٠','ثانية', '.']

CAMeL PoS: ['noun', 'verb', 'prep', 'noun','noun', 'noun_quant", 'digit', 'noun', 'punc']

H2: If ['foreign'] tag exists at sentence PoS, then it is more likely to be NFR.

The appearance of non-Arabic words often denotes techniques or programming languages, like (SQL, HTML...etc.). They are usually found in the description of NFR. If foreign words are existing in sentences in any sentence syntax, then it raises the likelihood that the sentence is a non-functional requirement. So, foreign words can distinguish non-functional requirements from functional based on their PoS tags. In order to check whether foreign words are present in the sentence, we have to look for all ['foreign'] tags and add the certainty factor of this heuristic to the NFR_CF for the certain sentence.

If this condition applies to a sentence, the certainty factor of H2 is added to

the NFR-CF of this sentence. NFR_CF = 80%

Some possible structures for requirements that show the locations of foreign words in Arabic sentences:

1. Verb + Subject + Object (1) | Object (2) | Object (3) -> Verb + (Noun | Pronoun | Forgan Word) + (Noun | Preposition + Noun |Preposition + Forgan Word | Forgan Word) + (Noun | Preposition + Noun | Preposition + Forgan Word | Forgan Word) + (Noun | Preposition + Noun | Preposition + Forgan Word | Forgan Word)

   CAMeL PoSTags:

   Verb + (noun | pron | forgan) + (noun | prep + noun | prep + forgan | forgan) + (noun | prep + noun | prep + forgan | forgan) + (noun | prep + noun | prep + forgan | forgan)

   Example: ".Windows XP يعمل البرنامج ضمن نظام التشغيل"

   A translation of the example: " The program works under Windows XP."

   CAMeL Tokens: ['يعمل','البرنامج','ضمن','نظام','التشغيل','Windows','XP','.']

   CAMeL PoS: ['verb', 'noun', 'noun', 'noun', 'forgan', 'forgan', 'punc']

2. Subject + Verb + Object (1) | Object (2) | Object (3) -> (Noun | Pronoun | Forgan Word) + Verb + (Noun | Preposition + Noun | Forgan Word | Preposition + Preposition) + (Noun | Preposition + Noun | Forgan Word | Preposition + Preposition) + (Noun | Preposition + Noun | Preposition + Forgan Word | Forgan Word)

   CAMeL PoS Tags:

(noun | pron | forgan) + verb + (noun | prep + noun | prep + forgan | forgan) + (noun | prep + noun | prep + forgan | forgan) + (noun | prep + noun | prep + forgan | forgan)

Example: ".Windows XP البرنامج يعمل ضمن نظام التشغيل"

A translation of the example: " The program works under Windows XP."

CAMeL Tokens: ['البرنامج','يعمل','ضمن','نظام','التشغيل','Windows','XP','.']

CAMeL PoS: ['noun','verb', 'noun', 'noun', 'forgan', 'forgan', 'punc']

We excluded from this heuristic the words that appear in parentheses, where they are usually explanatory words for some terms that pertain to the system field. In order to check the appearance of a foreign word between parentheses in the sentence, the first step is to look for this sequence of PoS tags ['punc', 'foreign', 'punc'] then we will not add the certainty factor of this heuristic to the NFR flag for the certain sentence. Such as the following example:

Example: ".(RBC) يقوم النظام بحساب عدد كريات الدم الحمراء"

A translation of the example: " The system counts the number of red blood cells (RBC)."

CAMeL Tokens: ['يقوم','النظام','ب','حساب','عدد','كريات','الدم','الحمراء','(','RBC',')','.']

CAMeL PoS: ['verb', 'noun', 'prep', 'noun', 'noun', 'noun_prop', 'noun', 'noun', 'punc', 'forgan', 'punc', 'punc']

H3: If ['adj'] or ['adv'] tag exists at sentence PoS, then it is more likely to be NFR.

The appearance of adjectives/adverbs indicates a high probability of NFR.

We propose a set of heuristics for verbal and nominal sentences as follows: If adjectives/adverbs are existing in sentences in any sentence syntax then it raises the likelihood that the sentence is a non-functional requirement whether it is written in numbers or letters. So, adjectives/adverbs may distinguish non-functional requirements from functional based on their Pos tags. In order to check whether adjectives/adverbs are present in the sentence, we have to look for all ['adj'] or ['adv'] tags and add the certainty factor of this heuristic to the NFR_CF for the certain sentence.

If this condition applies to a sentence, the certainty factor of H3 is added to the NFR-CF of this sentence. NFR_CF = 86%

Some possible structures for requirements that show the locations of adjectives/adverbs in Arabic sentences:

1. 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 الموقع جذابًا لجميع الجماهير."

   A translation of the example: "The site should be attractive to all audiences."

   CAMeL Tokens: ['يكون', 'الموقع', 'جذابا', 'ل', 'جميع', 'الجماهير', '.']

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

2. 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 | forgan) + Verb + (noun | prep + noun | noun + adv | adj) + (noun | prep + noun | noun + adv | adj) + (noun | prep + noun | noun + adv | adj)

   Example: "المنتج يستجيب بشكل سريع للحفاظ على البيانات المحدثة في الشاشة"

   A translation of the example: "The product responds quickly to keep the updated data in the screen"

   CAMeL Tokens: ['المنتج', 'يستجيب', 'ب', 'شكل', 'سريع', 'ل', 'لحفاظ', 'على', 'البيانات', 'المحدثة', 'في', 'الشاشة', '.']

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

H4: If any of lemmas of Tokens[sentence] exists at NFR keywords lookup table lemmas, then it is more likely to be NFR.

If the sentence contains keywords of non-functional requirements, that has been mentioned in table 4.4, and table 4.5. It raises the likelihood that the sentence is a non-functional requirement. We can find out whether the probability that this sentence is a non-functional requirement by returning the subjects/ object to their Lemma using CAMeL tools Lemmas then comparing them with the lemma of Keywords in table 4.4, and table 4.5. So, there are keywords that may distinguish non-functional requirements from functional ones.

If this condition applies to a sentence, the certainty factor of H4 is added to the NFR-CF of this sentence. NFR_CF = 90%

Some examples of requirements that contain NFR Keywords:

1. Example: "يجب أن يكون المنتج قابلا للصيانة."

   A translation of the example: "The product should be Maintainable."

   CAMeL Tokens: ['يجب ' 'أن','يكون','المنتج','قابلا','ل','لصيانة','.']

   CAMeL Lemma: ['وجب', 'أَنَ','كان','منتج','قابل','صيانة', '.']

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

2. Example: "يجب أن يتوافق التطبيق مع المتطلبات القانونية."

   A translation of the example: "The application should comply with legal requirements."

   CAMeL Tokens: ['يجب', 'ان','يتوافق','التطبيق','مع','المتطلبات','القانونية',' '.']

   CAMeL Lemma: ['وجب', 'أَنَ','توافق','تطبيق','مع','متطلبات','قانونيَ',' '.']

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

H5: If ['part_neg'] tag exists at sentence PoS, then it is more likely to be NFR.

Usually, non-functional requirements talk about determinants and limitations (Usually: security and performance), so the negative sentences may be classified closer to non-functional requirements. It can be determined by checking if any negative prefixes are present in the sentence. If negative prefixes are existing in sentences in any sentence syntax, then it raises the likelihood that

the sentence is a non-functional requirement whether it is written in numbers or letters. So, negative prefixes may distinguish non-functional requirements from functional based on their PoS tags. In order to check whether negative prefixes are present in the sentence, we have to look for all ['part_neg'] tags and add the certainty factor of this heuristic to the NFR flag for the certain sentence.

Negation in the Arabic language is the denial of something or its negation by something else that contradicts it. The negation method is used to negate the sentence, whether it is verbal or nominal, using one of the negation tools (لا، لما، لن، ليس، ما، لم).

If this condition applies to a sentence, the certainty factor of H5 is added to the NFR-CF of this sentence. NFR_CF = 82.66%

Some examples of requirements that contain negative prefixes:

1. Example: "لا يمكن انشاء حساب جديد الا بواسطة مسؤول النظام."

   A translation of the example: "A new account can only be created by the system administrator."

   CAMeL Tokens: [لا', 'يمكن','انشاء','حساب','جديد','الا','بواسطة','مسؤول', 'النظام','.']

   CAMeL PoS: [' part_neg ', 'verb', 'noun', 'noun', 'adj', 'part', 'prep', 'noun', 'noun', 'noun', 'punc']

2. Example: "يجب الحصول على النتائج في وقت لا يتجاوز ٣٠ ثانية."

   A translation of the example: "Results shall be obtained in no longer than 30 seconds."

   CAMeL Tokens: ['يجب', 'الحصول','على','النتائج','في','وقت','لا','يتجاوز','٣٠', 'ثانية', '.']

CAMeL PoS: ['verb', 'noun', 'prep', 'noun', 'prep', 'noun', 'part_neg', 'verb', 'digit', 'noun', 'punc']

H6: If the main actor is the "system", or one of its synonyms in Arabic, then it is more likely to be NFR.

If the system is the main actor in the requirement, indicating that the requirement. The system in Arabic could be: (النظام ، البرنامج ، التطبيق، الموقع). If the sentence is verbal then the main actor is the first subject after the main verb whereas (the main verb is the first verb in the sentence that is followed by a noun). This heuristic is presented as follows:

('verb', 'conj_sub',)Verb + 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)

If this condition applies to a sentence, the certainty factor of H6 is added to the NFR-CF of this sentence. . NFR_CF = 53.33%

Example:"يجب أن يكون المنتج متاحًا بنسبة ٩٩ % من الوقت."

A translation of the example: The product shall be available 99% of the time.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'المنتج','متاحا', 'بنسبة', '٩٩', '%', 'من', 'الوقت', '.']

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

As we noticed from our reading of the non-functional requirements that we reviewed, in many of them the actor is the system. When we were asked to evaluate this rule by experts, it was rated less than 75%, so we added some

conditions to it to be more accurate and credible, as follows:

1. In cases where the main verb in the sentence is “verb to be” that means it does not indicate an action, this is closer to non-functional requirements. For example, the system must be secure.

2. In cases where the main verb denotes movement and action, and the actor is the system, this is almost closer to functional requirements. Such as: that the system calculates the rates.

H7: We found some common structures for Functional Requirements. As we indicated in the previous heuristic there are correct structures of the Arabic sentence, we concluded through our study of the SRS documents there are certain structures are repeated in the functional requirements sentences, which we explain as follows:

- H7.1) ”أن” + Verb + Subject + Object (1) | Object (2) | Object (3) ->”أن” +”يكون” + (Noun | Pronoun) + ”قادرا” + (Noun | Preposition + Noun) + (Noun | Preposition + Noun)

  CAMeL PoS Tags:

  tokens [0] = [‘أن’] + verb + (noun | pron) + tokens [1] = [‘يكون’] + tokens [3] = [‘قادرا’] + (noun | prep + noun) + (noun | prep + noun)

  If this condition applies to a sentence, the certainty factor of H7.1 will add the FR_CF of this sentence. FR_CF = 88.33%

  Where the verb is يكون, and the first object is قادرا

  1. Example: ”أن يكون المشرف قادرا على تسجيل الطلاب .”

     A translation of the example: ”The supervisor should be able to enroll students.”

CAMeL Tokens: [’ان’, ’يكون’,’المشرف’,’قادرا’,’على’,’تسجيل’,’الطلاب’,’.’]

CAMeL PoS: : [’conj_sub’, ’verb’, ’noun’, ’adj’, ’prep’, ’noun’, ’noun’, ’punc’]

2. Example: ”أن يكون المستخدم قادرا على تغيير معلوماته.”

A translation of the example: ”The user is able to change his information.”

CAMeL Tokens: [’ان’, ’يكون’,’المستخدم’,’قادرا’,’على’,’تغيير’,’معلومات’,’ه’ ,’.’]

CAMeL PoS: [’conj_sub’, ’verb’, ’noun’, ’adj’, ’prep’, ’noun’, ’noun’,’pron’, ’punc’]

- H7.2) ”أن” + Subject +Verb + Object (1) | Object (2) | Object (3) -> ”أن” + ”يتمكن” + (Noun | Pronoun) + (Noun | Preposition + Noun) + (Noun | Preposition + Noun) + (Noun | Preposition + Noun)

  CAMeL PoS Tags:

  tokens [0] = [‘أن’] + tokens [1] = [‘يتمكن‘] + verb + (noun | pron) + (noun | prep + noun) + (noun | prep + noun)

  Where the verb is ’يتمكن.

  If this condition applies to a sentence, the certainty factor of H7.2 will add the FR_CF of this sentence.

  FR_CF = 88.33%

  1. Example: ”أن يتمكن الطالب من تسجيل مساق.”

     A translation of the example: ”The student should be able to register a course.”

     CAMeL Tokens: [’ان’, ’يتمكن’, ’الطالب’, ’من’, ’تسجيل’, ’مساق’, ’.’]

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

2. Example: "أن يتمكن المستخدم من إنشاء ملف."

   A translation of the example: "The user can create a file."

   CAMeL Tokens: [ان', يتمكن','المستخدم','من','إنشاء','ملف',' '.']

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

H8: If any of lemmas of sentence Token exists at FR keywords lookup table lemmas, then it is more likely to be FR.

Some words are repeated in the main verb of functional requirements sentences. The main verb is usually the first verb (PoS: verb) in the sentence after the main actor(PoS: noun).

Usually, the main verbs in functional requirements denote the movement "action". The main verb in the Arabic sentence can be known through the following linguistic structures:

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)

After getting the main verb from the sentence based on the sentence structure and on the output of complete tools, we convert the verb to its lemma using CAMeL tools and compare it from the lookup table shown below 4.2. We can find out whether the probability that this sentence is a functional requirement by returning the main verbs to their Lemma using CAMeL tools Lemmas then comparing them with the lemma of Keywords. So, there are

Table 4.2: Functional Main Verb Keywords.

English Verbs	Arabic Verbs
Add	يضيف
Register	يسجل
Modify	يعدل
Update	يحدث
Filling	يعبئ
Activate	يفعل
Display	يعرض
Delete	يحذف
Calculate	يحسب
Create	ينشئ
Search	يبحث
Choose	يختار
Send	يرسل

keywords that may distinguish non-functional requirements from functional ones.

If this condition applies to a sentence, the certainty factor of H8 is added to the FR_CF of this sentence. FR_CF = 83.33%

1. Example: "أن ينشئ المستخدم رسالة نصية"

   A translation of the example: "The user should creates a text message."

   CAMeL Tokens: ['ان', 'ينشئ', 'المستخدم', 'رسالة', 'نصية', '.']

   CAMeL Lemma: ['أَنَّ', 'أَنْشَأَ', 'مُسْتَخْدِم', 'رِسالَة', 'نَصّ', '.']

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

2. Example: "أن يختار الطالب المساقات التي يريد تسجيلها" A translation of the example: "The student chooses the courses he wants to register."

Table 4.3: Functional Requirements Keywords.

English Verbs	Arabic Verbs
Adding	اضافة
Registration	تسجيل
Modification	تعديل
Update	تحديث
Filling	تعبئة
Activation	تفعيل
Displays	عرض
Delete	حذف
Calculation	حساب
Creation	انشاء
Search	بحث
Selection	اختيار
Sending	ارسال

CAMeL Tokens: ['ان', 'يختار', 'الطالب', 'المساقات', 'التي', 'يريد', 'تسجيل', 'ها', '.']

CAMeL Lemma: ['أنّ', 'اخْتار', 'طالِب', 'المساقات', 'الَّذِي', 'أَرَدَى', 'تَسْجِيل', '.']

CAMeL PoS: ['conj_sub', 'verb', 'noun', 'noun_prop', 'pron_rel', 'verb', 'noun', 'pron', 'punc']

Also, this action can be contained in the form of words (PoS: noun) in some sentence structures. These words are usually found after the preposition in the sentence. After getting any of these words from the sentence based on the sentence structure and on the output of complete tools, we convert it to its lemma using CAMeL tools and compare it from the lookup table shown below 4.3.

1. Example: "أن يكون المستخدم قادرًا على إضافة صديق"

   A translation of the example: "User should be able to add a friend."

CAMeL Tokens: ['أن', 'يكون', 'المستخدم', 'قادرا', 'على', 'اضافة', 'صديق', '.']

CAMeL Lemma: ['أَنَّ', 'كان', 'المستخدم', 'قادِر', 'على', 'إِضافَة', 'صِدِّيق', '.']

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

2. Example: "أن يتمكن المشرف من تفعيل حساب طالب"

A translation of the example: "The supervisor should be able to activate a student account."

CAMeL Tokens: ['أن', 'يتمكن', 'المشرف', 'من', 'تفعيل', 'حساب', 'طالب', '.']

CAMeL Lemma: ['أَنَّ', 'تَمَكَّن', 'مُشَرِّف', 'مِن', 'تَفْعِيل', 'حِساب', 'طالَب', '.']

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

H9: If the sentence is a conditional sentence, it means that there is highly probability that it is FR.

The sentence is more likely to be FR if its structure as the following:

Subordinating Conjunction + (Verb | Negative Particle +Verb) + (Connective Particle + Pseudo Verb) | Future Particle | (Response Conditional+ Future Particle) + verbal sentence.

If the sentence is a conditional sentence, means that it is highly probability of FR. We propose a set of heuristics for conditional sentences as follows:

Conditional sentences could be:

- Proof sentence.
- Negation sentence.

Conditional Arabic sentence's structure is:

1Conditional Particle + 2Conditional sentence + 3Answer Particle + 4Conditional Answer

1. Conditional particle: There are two common condition particles in Arabic Language ("إذا","لو"). These tags are (subordinating conjunction) in CAMeL tools ['conj'].
2. Conditional sentence: A conditional sentence is a verbal sentence. There are two types of conditional sentences: proof and negation sentences.
    - Proof sentence contests of condition particle followed by conditional sentence directly without existing of negation particle (negative particle " لم")
    - Negation sentence has the negation particle (negative particle " لم") after the conditional particle.
3. Answer Particle: Answer particle is an adverb for the condition answer. Answer particles in Arabic language are:"فإن", "سوف", "فسوف". The tags for them are: ("ف" connective particle +"إن" Pseudo verb) , ("سوف" Future particle) , ("ف" Response conditional)
4. Conditional answer: The conditional answer is a verbal sentence.

Conditional sentences Tags:

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))

If this condition applies to a sentence, the certainty factor of H9 is added the FR_CF of this sentence. FR_CF = 76.66%

1. Example: "اذا كان عدد الطلاب في الشعبة أقل من الحد الأدنى سيتم حذفها."

   A translation of the example: "If there are no registered students, the supervisor will be able to delete the section."

   CAMeL Tokens: ['اذا', 'كان', 'عدد', 'الطلاب', 'في', 'الشعبة', 'أقل', 'من', 'الحد', 'الادنى', 'س', 'يتم', 'حذف', 'ها', '.']

   CAMeL PoS: ['conj', 'verb', 'noun', 'noun', 'prep', 'noun', 'noun', 'prep', 'noun', 'noun', 'part_fut', 'verb', 'noun', 'pron', 'punc']

2. Example: "سيتمكن الطالب من رؤية علامته إذا قام المسؤول باظهارها."

   A translation of the example: "The student will be able to see his mark if the supervisor shows it."

   CAMeL Tokens: ['س', 'يتمكن', 'الطالب', 'من', 'رؤية', 'علامات', 'ه', 'اذا', 'قام', 'المسؤول', 'ب', 'اظهار', 'ها', '.']

   CAMeL PoS: ['part_fut', 'verb', 'noun', 'prep', 'noun', 'noun', 'pron','conj', 'verb', 'adj', 'prep', 'noun', 'pron', 'punc']

4.4 The Novel Approach for Classification of Arabic User Requirements

The following algorithmic steps represent the classification of the Arabic user requirements into functional and non-functional requirements based on the presented heuristics.

Algorithm 1 Algorithm of Applying Heursticis

Input: Unclassified Arabic User Requirements
Output: Classified Requirements
FR_Score [] = 0
NFR_Score [] = 0
FR_CF [] = 0
NFR_CF[] = 0
Tokens = []
PoS_Tags = []
for all Unclassified Arabic User Requirements statements
Step 1: Normalize the Arabic user requirements using CAMeL tools.
Step 2: Tokenize the Arabic user requirements using CAMeL tools.
Step 3: Generate the PoS tags of the Arabic user requirements tokens using CAMeL tools.
Step 4: Apply proposed heuristics from H1 to H9:
if ['digit'] or ['noun_num'] tag exists at PoS_Tags[sentence] **then**
NFR_Score[sentence] = +1
NFR_CF [sentence] = Maximum (NFR_CF [sentence], 0.875)
end if
if ['foreign'] tag exists in PoS_Tags[sentence] **then**
NFR_Score[sentence] = +1
NFR_CF [sentence] = Maximum (NFR_CF [sentence], 0.90)
end if
if ['adj'] or ['adv'] tag exists at PoS_Tags[sentence] **then**
NFR_Score[sentence] = +1
NFR_CF [sentence] = Maximum (NFR_CF [sentence], 0.86)
end if
if any of lemmas of Tokens[sentence] existing at NFR keywords lookup table lemmas **then**
NFR_Score[sentence] = +1
NFR_CF [sentence] = Maximum (NFR_CF [sentence], 0.90)
end if

if ['part_neg'] tag exists at PoS_Tags[sentence] **then**
NFR_Score[sentence] = +1
NFR_CF [sentence] = Maximum (NFR_CF [sentence], 0. 8266)
end if
if the sentence is verbal and the main actor is the "system" & the ['verb'] Tokens is 'verb to be' **then**
NFR_Score[sentence] = +1
NFR_CF [sentence] = Maximum (NFR_CF [sentence], 0.5333)
end if
if the requirements are at one of the Functional Requirements structures **then**
FR_Score[sentence] = +1
FR_CF [sentence] = Maximum (FR_CF [sentence], 0.88.33)
end if
if any of lemmas of Tokens[sentence] existing at FR keywords lookup table lemmas. **then**
FR_Score[sentence] = +1
FR_CF [sentence] = Maximum (FR_CF [sentence], 0. 8333)
end if
if the sentence is a conditional sentence, means that it is highly probability of Functional Requirements. **then**
FR_score[sentence] = +1
FR_CF[sentence] = Maximum (FR_CF[sentence], 0. 7666)
end if
end for
Step 5: Compare each sentence's Scores and CFs to determine the class it belongs to:
if (FR_Score[sentence] > NFR_Score[sentence]) **then**
Class[Sentence] = FR
else if (FR_Score[sentence] < NFR_Score[sentence]) **then**
Class [Sentence] = NFR
else if (FR_Score[sentence] == NFR_Score[sentence]) **then**
if (FR_CF[sentence] > NFR_CF[sentence]) **then**
Class[Sentence] = FR
else if (FR_CF[sentence] < NFR_CF[sentence]) **then**
Class[Sentence] = NFR
else if (FR_CF[sentence] == NFR_CF[sentence]) **then**
Class [Sentence] = Nan
end if
end if

4.5 Case Study

In this section, we illustrate the proposed approach and show how we classify Arabic user requirements. We chose one of the projects on the Internet to test our approach. We chose an SRS document written in English [44], and then we request a translation expert to translate the functional and non-functional requirements into Arabic. We chose 10 Functional requirements by selecting the first 10 odd numbers, then we chose 10 non-functional requirements in the same way to be our benchmark 4.6.

The chosen SRS is for an Android Mobile Application project. The main idea of this project is to provide a new messaging system named location-based messaging system and a new social platform that gives people a chance to meet new people with the same interests.

This project has two major features: The first one is a location-based messaging system. It enables the user to see the location of his/her friend and message with that friend on the same interface. The second major feature is event-based which enables the user to create events and share posts with the locations, photos, and contents.

Table 4.6: Case Study

	Sentences in Arabic	**Sentences in English**
Functional	يجب أن يكون المستخدم قادرًا على لتحقق من رقم هاتفه قبل عملية التسجيل الفعلية.	User shall be able to verify his/her phone number before actual registration process.

يجب أن يكون المستخدم قادرًا على حذف حسابه من نظام التطبيق.	User shall be able to delete his account from system of application.
يجب أن يكون المستخدم قادرًا على حذف صديق من أصدقائه.	User shall be able to delete friend in his friends.
يجب أن يكون المستخدم قادرًا على رؤية موقع الأصدقاء على الخريطة الرئيسية.	User shall be able to see location of friends on main map.
يجب أن يكون المستخدم قادرًا على تحديد المسافة كقطر لرؤية صديق في منطقة معينة.	User shall be able to select the distance as diameter in order to see friend in certain area.
يجب أن يكون المستخدم قادرًا على الدردشة مع صديق واحد بينما يرى المستخدم موقعه و موقع صديقه.	User shall be able to chat with one friend while user is seeing his location and his friend's location.

	يجب أن يكون المستخدم قادرًا على اختيار الأصدقاء لمشاركة الحدث معهم فقط.	User shall be able to select friends in order to share event only with them.
	يجب أن يكون المستخدم قادرًا على رؤية الأحداث الخاصة التي تمت مشاركتها معه على الخريطة.	User shall be able to see private events that shared with him on map.
	يجب أن يكون المستخدم قادرًا على رؤية جميع الأحداث المميزة على الخريطة.	User shall be able to see all privileged events on map.
	يجب أن يكون المستخدم قادرًا على إرسال طلب إلى مبتكر الحدث العام للانضمام إلى الحدث.	User shall be able to send a request to creator of public event for joining event.
Non_Functional	يجب أن يكون التطبيق قادرا على دعم ما لا يقل عن ١٠٠٠٠٠٠ مستخدم في وقت واحد.	The app shall be able to support at least 100.000 simultaneous users.

All of the functions such as registration, sending messages, receiving messages, seeing locations of friends on map, seeing events on map shall perform in less than 3 seconds.	يجب أداء جميع الوظائف مثل التسجيل وإرسال الرسائل واستلام الرسائل ورؤية مواقع الأصدقاء على الخريطة ورؤية الأحداث على الخريطة في أقل من ٣ ثوانٍ.
Other applications on user's phone should not reach and manipulate stored data in phone of user.	يجب ألا تصل التطبيقات الأخرى الموجودة على هاتف المستخدم إلى البيانات المخزنة في هاتف المستخدم وتتلاعب بها.
Sent and received data should be transferred via HTTP connection.	يجب نقل البيانات المرسلة والمستلمة عبر اتصال *HTTP*.
Password shall be at least 6 characters.	يجب أن تتكون كلمة المرور من ٦ أحرف على الأقل.
Users should not reach and manipulate stored data of other users in database.	لا يسمح للمستخدمين الوصول إلى البيانات المخزنة للمستخدمين الآخرين في قاعدة البيانات ومعالجتها.

	يجب أن يكون النظام متاحًا لمدة ٧ أيام و ٢٤ ساعة.	The system should be available for 7 days and 24 hours.
	لتقليل التعقيد ، وجعل النظام قابلاً للتتبع أثناء تطوير النظام، يجب استخدام نظام التحكم في الإصدار مثل *gitlab* .	To reduce complexity and make the system traceable, a version control system such as gitlab should be used.
	يجب أن يكون إصدار نظام التشغيل *Android* على الجهاز أكبر من ٣.	The version of Android Operating system on device should bigger than 3.
	يجب أن يحافظ هذا النظام على تحديث قاعدة البيانات.	This system should keep the database updated.

1. Arabic Sentence:

   يجب أن يكون المستخدم قادرًا على التحقق من رقم هاتفه قبل عملية التسجيل الفعلية.

   CAMeL Tokens: ['يجب', 'ان', 'يكون', 'المستخدم', 'قادرا', 'على', 'التحقق', 'من', 'رقم', 'هاتف', 'ه', 'قبل', 'عملية', 'التسجيل', 'الفعلية','.']

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

2. Arabic Sentence:

يجب أن يكون المستخدم قادرًا على حذف حسابه من نظام التطبيق.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'المستخدم' 'قادرا', 'على', 'حذف', 'حساب' 'ه', 'من', 'نظام', 'التطبيق' ,'.']

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

3. Arabic Sentence:

يجب أن يكون المستخدم قادرًا على حذف صديق من أصدقائه.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'المستخدم' 'قادرا', 'على', 'حذف', 'صديق' 'من', 'اصدقاء' 'ه' '.']

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

4. Arabic Sentence:

يجب أن يكون المستخدم قادرًا على رؤية موقع الأصدقاء على الخريطة الرئيسية.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'المستخدم', 'قادرا', 'على', 'رؤية', 'موقع' 'الاصدقاء', 'على' 'الخارطة', 'الرئيسية', '.']

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

5. Arabic Sentence:

يجب أن يكون المستخدم قادرًا على تحديد المسافة كقطر لرؤية صديق في منطقة معينة.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'المستخدم' 'قادرا', 'على', 'تحديد', 'المسافة' 'ك', 'قطر', 'ل', 'رؤية', 'صديق', 'في', 'منطقة', 'معينة' '.']

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

'noun', 'prep','noun', 'prep', 'noun', 'noun', 'prep', 'noun', 'adj', 'punc']

6. Arabic Sentence:

يجب أن يكون المستخدم قادرًا على الدردشة مع صديق واحد بينما يرى المستخدم موقعه و موقع صديقه.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'المستخدم', 'قادرا', 'على', 'الدردشة', 'مع', 'صديق', 'واحد', 'بينما', 'يرى', 'المستخدم', 'موقع', 'ه', 'و', 'موقع', 'صديق', 'ه', '.']

CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'adj', 'prep', 'noun', 'prep', 'noun_prop', 'noun', 'conj_sub', 'verb', 'noun', 'noun', 'pron','conj', 'noun', 'noun', 'pron', 'punc']

7. Arabic Sentence:

يجب أن يكون المستخدم قادرًا على اختيار الأصدقاء لمشاركة الحدث معهم فقط.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'المستخدم', 'قادرا', 'على', 'اختيار', 'الاصدقاء', 'ل', 'مشاركة', 'الحديث', 'مع', 'هم', 'فقط', '.']

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

8. Arabic Sentence:

يجب أن يكون المستخدم قادرًا على رؤية الأحداث الخاصة التي تمت مشاركتها معه على الخريطة.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'المستخدم', 'قادرا', 'على', 'رؤية', 'الاحداث', 'الخاصة', 'التي', 'تمت', 'مشاركت', 'ها', 'مع', 'ه', 'على', 'الخارطة', '.']

CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'adj', 'prep', 'noun', 'noun', 'adj', 'pron_rel', 'verb', 'noun', 'pron', 'prep', 'pron', prep',

'noun', 'punc']

9. Arabic Sentence:

يجب أن يكون المستخدم قادرًا على رؤية جميع الأحداث المميزة على الخريطة.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'المستخدم' , 'قادرا', 'على', 'رؤية',
'جميع', 'الاحداث', 'المميزة', 'على', 'الخارطة', '.']

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

10. Arabic Sentence:

يجب أن يكون المستخدم قادرًا على إرسال طلب إلى مبتكر الحدث العام للانضمام إلى الحدث.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'المستخدم', 'قادرا', 'على', 'ارسال',
'طلب', 'الى', 'مبتكر','الحدث', 'العام', 'ل', 'الانضمام', 'الى', 'الحدث', '.']

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

11. Arabic Sentence:

يجب أن يكون التطبيق قادرا على دعم ما لا يقل عن ١٠٠٠٠٠٠ مستخدم في وقت واحد.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'التطبيق' 'قادرا', 'على', 'دعم', 'ما',
'لا', 'يقل', 'عن' '١٠٠', '٠٠٠', '.', '١٠٠', 'مستخدم', 'في', 'وقت', 'واحد', '.']

CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'adj', 'prep', 'noun',
'pron_rel', 'part_neg', 'verb', 'prep', 'digit', 'punc', 'digit', 'adj', 'prep',
'noun', 'noun', 'punc']

12. Arabic Sentence:

يجب أداء جميع الوظائف مثل التسجيل و إرسال الرسائل واستلام الرسائل و

رؤية مواقع الأصدقاء على الخريطة و رؤية الأحداث على الخريطة في أقل من ٣ ثوانٍ.

CAMeL Tokens: ['يجب', 'اداء', 'جميع', 'الوظائف', 'مثل', 'التسجيل', 'و', 'رؤية', 'الخريطة', 'و', 'رؤية', 'الاصدقاء', 'مواقع', 'على', 'الخريطة', 'و', 'الرسائل', 'ارسال', 'ثوانٍ', '٣', 'من', 'اقل', 'في', 'الخريطة', 'على', 'الاحداث', '.']

CAMeL PoS: ['verb', 'noun', 'noun', 'noun', 'noun', 'noun', 'conj', 'noun', 'noun', 'conj, 'noun', 'noun', 'conj', 'noun', 'noun', 'noun', 'prep', 'noun', 'conj', 'noun', 'noun', 'prep', 'noun', 'prep', 'noun', 'prep', 'digit', 'noun', 'punc']

13. CAMeL PoS:Arabic Sentence:

يجب ألا تصل التطبيقات الأخرى الموجودة على هاتف المستخدم إلى البيانات المخزنة في هاتف المستخدم و تتلاعب بها.

CAMeL Tokens: ['يجب', 'الا', 'تصل', 'التطبيقات' 'الاخرى', 'الموجودة', 'على', 'هاتف', 'المستخدم', 'الى', 'البيانات', 'المخزنة', 'في', 'هاتف' 'المستخدم', 'و', 'تتلاعب', 'ب', 'ها', '.']

CAMeL PoS: ['verb', 'part_neg', 'verb', 'noun', 'adj', 'noun', 'prep', 'noun', 'noun', 'prep', 'noun', 'noun_prop', 'prep', 'noun', 'noun', 'conj', 'verb', 'prep', 'pron', 'punc']

14. Arabic Sentence:

يجب نقل البيانات المرسلة و المستلمة عبر اتصال *HTTP*.

CAMeL Tokens: ['يجب', 'نقل', 'البيانات', 'المرسلة' ,'و', 'المستلمة', 'عبر', 'اتصال', 'HTTP' ,'.']

CAMeL PoS: ['verb', 'noun', 'noun', 'noun', 'conj', 'noun', 'noun', 'noun', 'foreign', 'punc']

15. Arabic Sentence:

يجب أن تتكون كلمة المرور من ٦ أحرف على الأقل.

CAMeL Tokens: ['يجب', 'ان', 'تتكون', 'كملة', 'المرور', 'من', '٦', 'احرف', 'على', 'الاقل', '.']

CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'noun', 'prep', 'digit', 'noun', 'prep', 'noun', 'punc']

16. Arabic Sentence:

لا يسمح للمستخدمين الوصول إلى البيانات المخزنة للمستخدمين الآخرين في قاعدة البيانات ومعالجتها.

CAMeL Tokens: ['لا', 'يسمح', 'ل', 'لمستخدمين', 'الوصول', 'الى', 'البيانات', 'المخزنة', 'ل', 'لمستخدمين', 'الاخرين', 'في', 'قاعدة', 'البيانات', 'و', 'معالجت', 'ها', '.']

CAMeL PoS: ['part_neg', 'verb', 'prep', 'noun', 'noun', 'prep', 'noun', 'noun_prop', 'prep', 'noun', 'adj', 'prep', 'noun_prop', 'noun', 'con', 'noun','pron', 'punc']

17. Arabic Sentence:

يجب أن يكون النظام متاحًا لمدة ٧ أيام و ٢٤ ساعة.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'النظام', 'متاحا', 'ل', 'مدة', '٧', 'ايام', 'و', '٢٤', 'ساعة', '.']

CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'noun', 'prep', 'noun', 'digit', 'noun', 'conj', 'digit', 'noun', 'punc']

18. Arabic Sentence:

لتقليل التعقيد و جعل النظام قابلاً للتتبع أثناء تطوير النظام، يجب استخدام نظام التحكم في الإصدار مثل *gitlab* .

CAMeL Tokens: ['ل', 'تقليل', 'التعقيد', 'و', 'جعل', 'النظام', 'قابلا', 'ل',

'في', 'التحكم', 'نظام', 'استخدام', 'يجب', '،', 'النظام', 'تطوير', 'اثناء', 'لتتبع', 'الاصدار', 'مثل', 'gitlab', '.']

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

19. Arabic Sentence:

يجب أن يكون إصدار نظام التشغيل *Android* على الجهاز أكبر من ٣.

CAMeL Tokens: ['يجب', 'ان', 'يكون', 'اصدار', 'نظام', 'التشغيل', 'Android', 'على', 'الجهاز', 'اكبر', 'من', '٣', '.']

CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'noun', 'noun', 'foreign', 'prep', 'noun', 'adv', 'prep', 'digit' 'punc']

20. Arabic Sentence:

يجب أن يحافظ هذا النظام على تحديث قاعدة البيانات.

CAMeL Tokens: ['يجب', 'ان', 'يحافظ', 'هذه', 'النظام', 'على', 'تحديث', 'قاعدة', 'البيانات', '.']

CAMeL PoS: ['verb', 'conj_sub', 'verb', 'pron_dem', 'noun', 'prep', 'noun', 'noun_prop', 'noun', 'punc']

Table 4.7 represents the result of analyzing the case study requirements tags based on the proposed heuristics to classify requirements. It contains the heuristics applied to each requirement, numbers of heuristics support the possibility that the requirement is functional which is represented by FR_Score, and numbers of heuristics support the possibility that the requirement is nun-functional which is represented by NFR_Score. Also, it contains the certainty factor of being a functional requirement and the certainty factor of being a non-functional requirement for each requirement. The last

column represents the class of each requirement depending on the proposed heuristics.

Table 4.7: Case Study Classification Results Based on the Proposed Approach.

Req#	Heuristic/s Applied To	FR_Score	NFR_Score	FR_CF	NFR_CF	Class
Req#1	H3, H7, H8	2	1	88.3	86	FR
Req#2	H3, H7, H8	2	1	88.3	86	FR
Req#3	H3, H7, H8	2	1	88.3	86	FR
Req#4	H3, H7	1	1	88.3	86	FR
Req#5	H3, H7	1	1	88.3	86	FR
Req#6	H3, H7	1	1	88.3	86	FR
Req#7	H3, H7, H8	2	1	88.3	86	FR
Req#8	H3, H7	1	1	88.3	86	FR
Req#9	H3, H7	1	1	88.3	86	FR
Req#10	H3, H7, H8	2	1	88.3	86	FR
Req#11	H1, H3, H5, H6, H7	1	4	88.3	86	NFR
Req#12	H1, H4	0	2	0	87.5	NFR
Req#13	H3, H5	0	2	0	86	NFR
Req#14	H3	0	1	0	80	NFR
Req#15	H1	0	1	0	78.33	NFR
Req#16	H3, H4, H5	0	3	0	90	NFR
Req#17	H1	0	1	0	78.33	NFR
Req#18	H2, H3	0	2	0	90	NFR
Req#19	H1, H2, H3	0	3	0	86	NFR
Req#20	H8	1	0	83.33	0	FR

Table 4.4: NFR Keywords (1)

English	Arabic	English	Arabic
Accessibility	إمكانية الوصول	Auditability	تدقيق
Adaptability	تكيف	Visibility	وضوح
Affordability	تحمل التكاليف	Compatibility	توافق
Availability	توفر	Compatibility	إيجاز
Capacity	سعة	Consistency	تناسق
Understandability	فهم	Coordination Cost	تكلفة التنسيق
Confidentiality	سرية	Cost	تكلفة
Controllability	قدرة على التحكم	Development Cost	تكلفة التطوير
Correctness	صحة	Diversity	تنوع
Space Performance	أداء مساحة البيانات	Evolvability	قابلية التطور
Dependability	اعتمادية	Flexibility	مرونة
Enhanceability	تحسين	Maintainability	قابلية الصيانة
Extensibility	تمدد	Maintenance Time	وقت الصيانة
Interoperability	توافقية	Measurability	قياس
Learnability	تعلم	Performance	أداء
Maintenance	صيانة	Plasticity	ليونة
Performance	أداء	Project Stability	استقرار المشروع
Modifiability	تعديل	Recoverability	استرداد
Observability	قابلية الملاحظة	Reliability	موثوقية
Productivity	إنتاجية	Replicability	قابلية التكرار
Promptness	سرعة	Reusability	إعادة الاستخدام
Reconfigurability	إعادة التكوين	Scalability	قابلية التوسع
Responsiveness	إستجابة	Surety	ضمان
Safety	سلامة	Sustainability	استدامة
Security	الأمان	Throughput	الإنتاجية
Simplicity	بساطة	Transparency	شفافية
Stability	استقرار	Usability	سهولة الاستخدام
Supportability	دعم	Maintainability	قابلية التعديل
Testing Time	وقت الاختبار	Integrity	تكامل
Timeliness	توقيت	Execution Cost	تكلفة التنفيذ
Trainability	قابلية التدريب	Fault-Tolerance	تسامح مع الخطأ
Understandability	قابلية للفهم	Mobility	إمكانية التنقل
Validity	صلاحية	Robustness	متانة
Sensitivity	حساسية	Response Time	وقت الاستجابة

Table 4.5: NFR Keywords (2)

English	Arabic	English	Arabic
Completeness	اكتمال	Space	مساحة
Accuracy	دقة	Time	زمن
Perturbation	اضطراب	Memory	ذاكرة
Virus	فايروس	Storage	تخزين
Access	وصول	Response	استجابة
Authorization	تفويض	Index	فهرس
Card	بطاقة	Compress	ضغط
Key	مفتاح	Uncompressed	فك الضغط
Password	كلمه السر المرور	Runtime	مدة العرض
Noise	ضوضاء	Perform	نفذ
Fixing	اصلاح	Execute	ينفذ
Early	مبكرا	Alarm	إنذار
Late	متأخر	Encryption	تشفير
Appear	يظهر	Year	عام
Update	تحديث	Train	يدرب
Interface	واجهه	Hour	ساعة
Achieve	حقق	Color	لون
Regular	منتظم	Prevent	يمنع
Interface	واجهه	Fast	سريع
Maintain	صيانة	Look	مظهر
Response	يستجيب	Minute	دقيقة
Handle	يتعامل	Expect	توقع
Easy	سهل	Ensure	تأكد
Standard	اساسي	Promote	ترويج
Release	اصدار	Understand	فهم
Server	خادم	Legal	قانوني
Author	مؤلف	Browser	متصفح
Appeal	جاذبية	Process	معالجة
Secure	امن	Successfully	نجاح
Estimate	تقدير	Downtime	توقف
Scheme	مخطط	Law	قانون
Logo	شعار	Window	نافذة
Budget	ميزانية	Complete	مكتمل
Launch	يطلق	Defect	خلل
Develop	طور	Feel	يشعر

Chapter 5

Evaluation

This chapter presents and analyzes the experimental results of applying the proposed approach.

5.1 Evaluation Methodology

In this chapter, we present the evaluation of our approach. The purpose of the evaluation is to calculate the accuracy of our approach and compare the result with those derived by graduate and undergraduate students and check if our approach is better than students' requirements classification. We use three threshold types of discriminator metrics to evaluate our approach. These metrics are as follows [45]:

1. Accuracy (acc).
2. Precision (p).
3. Recall (r).

Confusion Matrix is also used as a performance measurement for classification problems. It is a table with combinations of predicted and actual values.

Table 5.1: Confusion Matrix

	Actual Positive Class	Actual Negative Class
Predicted Positive Class	True Positive (TP)	False Negative (FN)
Predicted Negative Class	False positive (FP)	True Negative (TN)

The row of the table represents the predicted class, while the column represents the actual class. From this confusion matrix, TP and TN denote the number of positive and negative instances that are correctly classified. Meanwhile, FP and FN denote the number of misclassified negative and positive instances, respectively.

- Accuracy(acc): In general, it measures the ratio of correct predictions over the total number of instances evaluated. In our approach it measures the ratio of correct requirements statement classified true over the total number of requirements.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{5.1}$$

- Precision(P): In general, Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. In our model, we measure precision for both classes.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

  PFR: is the ratio of correctly predicted FR statements to the total predicted FR statements PNFR: is the ratio of correctly predicted NFR statements to the total predicted NFR statements.

- Recall (R): In general, it is the ratio of correctly predicted positive observations to all observations in the actual class. In our model, we

measure recall for both classes.

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

PFR: is the ratio of correctly predicted FR statements to all observations in FR class. PNFR: is the ratio of correctly predicted NFR statements to all observations in NFR class.

5.2 Experiments

User requirements for a set of cases are classified in three ways: using the proposed approach of this thesis, by a set of graduate students, and a set of undergraduate students. The comparison of the results in all cases has been done and the accuracy measurements have been calculated.

We chose three SRS for open-source projects to test our approach. The SRS documents are written in English. So, we request a translation expert to translate the project's user requirements into Arabic. We chose ten Functional requirements by selecting the first ten odd numbers, then we chose ten non-functional requirements in the same way to be our case studies. Then we asked a software engineering expert to reclassify the requirements. To adopt his classification as a benchmark for our experiments.

Three case studies were given to five graduate students in the master of the informatics program with good knowledge of software requirements. We also gave the first case study to ten excellent undergraduate students from the College of Information Technology and Computer Engineering at Palestine Polytechnic University. They are at third or fourth academic levels (They have completed a software engineering course).

We created Google forms for both graduate students and undergraduate students in which there is a brief explanation and twenty unclassified requirements for each project. They can choose between three options (functional, non-functional, or I don't know).

To evaluate our approach, we wrote a Python code using the CAMeL tools for pre-processing and part of speech tag generation. Our experiments were implemented using CAMeL Tools 1.3.1, and Python 3.6.7, under ubuntu 20.04 LTS.

Each project's requirements are stored as a CSV file. It was automatically categorized using the code, and the result was printed on another CSV file. The code classifies the requirement into FR or NFR and if it cannot determine any requirement class, it leaves it to user intervention. Then the evaluation of the approach was done based on the benchmark.

5.3 Evaluation Results

The results obtained from the graduate students, undergraduate students, our approach, and the benchmark including the percentage of correct, incorrect, and indeterminate classified requirements for the graduate students, undergraduate students, benchmark, and our approach are summarized in Tables 5.2, 5.3 and 5.4.

Table 5.2: Percentage of Correct Classified Requirements

Correct	Benchmark	Undergraduate	Graduate	Our Approach
Functional	100%	85%	94%	100%
Non-Functional	100%	82%	90%	90%

As presented in Table 5.2, the percentage of correct functional and non-functional classified requirements for undergraduate students are 85%, 84%,

Table 5.3: Percentage of Incorrect classified requirements

Incorrect	Benchmark	Undergraduate	Graduate	Our Approach
Functional	0%	7%	4%	0%
Non-Functional	0%	10%	6%	10%

Table 5.4: Percentage of Unclassified classified requirements

Unclassified	Benchmark	Undergraduate	Graduate	Our Approach
Functional	0%	7%	4%	0%
Non-Functional	0%	8%	4%	0%

respectively. And for graduate students are 94%, 90%, respectively. Our proposed classify the functional requirements correctly as 100% and the non-functional requirements at 90% in this case study.

As shown in Table 5.3, the percentage of incorrect correct functional and non-functional classified requirements for undergraduate students are 7%, 10%, respectively. And for graduate students are 4%, 6%, respectively. Our proposed classify the functional requirements incorrectly as 0% and the non-functional requirements at 10% in this case study.

As shown in Table 5.4, the percentage of the requirements that are unclassified functional and non-functional requirements for undergraduate students are 8%, 8%, respectively. And for graduate students are 2%, 4%, respectively. Our proposed approach Classify the functional requirements Incorrectly as 0% and the non-functional requirements at 0% in this case study.

Table 5.5 shows the evaluation of measure metrics: accuracy, precision, and recall. Here, we note that our approach is higher evaluation results as compared to the results of activity diagrams of graduate students.

In addition to the previous case study, we solved two different case studies of various lengths [46][47]. The results of all the case studies were used to

Table 5.5: Measure Metrics Results

Metrics	Benchmark	Undergraduate	Graduate	Our Approach
Accuracy	100%	83.5%	93%	95%
Precision (FR)	100%	82%	90%	91%
Precision (NFR)	100%	84%	93.8%	100%
Recall (FR)	100%	85%	94%	100%
Recall (NF)	100%	82%	90%	90%

calculate accuracy, precision, and recall shown in table 5.6.

Generally, we concluded from these results, that our approach is a better method to classify user requirements written in Arabic into functional and non-functional than graduate and undergraduate students. Also from the results, we know that our approach is more accurate in determining non-functional requirements than the functional ones. Whereas the average recall of functional is equal to 83.33% while the average recall of non-functional is equal to 93.33%.

Table 5.6: Evaluation Result

	Benchmark					Graduate					Our Approach				
	Acc.	**P. (FR)**	**P. (NFR)**	**R. (FR)**	**R. (FR)**	**Acc.**	**P. (FR)**	**P. (NFR)**	**R. (FR)**	**R. (FR)**	**Acc.**	**P. (FR)**	**P. (NFR)**	**R. (FR)**	**R. (FR)**
Case#1	100%	100%	100%	100%	100%	93%	90%	93.8%	94%	90%	95%	91%	100%	100%	90%
Case#2	100%	100%	100%	100%	100%	84%	81%	86%	88%	80%	85%	100%	76%	70%	100%
Case#3	100%	100%	100%	100%	100%	87%	93%	82%	80%	94%	90%	88%	81%	80%	90%
Average	100%	100%	100%	100%	100%	88.6%	88%	87.2%	87.3%	88%	90%	93%	85.66%	83.33%	93.33%

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we proposed a novel approach for semi-automated classification of user requirements written in Arabic using natural language processing tools (CAMeL tools). A set of heuristics is presented to classify Arabic user requirements. These heuristics use the tokens, PoS tags, and lemmas produced by CAMeL tools. These outputs are used in different ways throughout each heuristic to classify requirements into functional and non-functional.

We implement the proposed approach using Python code and CAMeL tools API. The proposed is very useful for software engineers who are responsible for writing and analyzing Arabic user requirements. This research aims to help software engineers in the analysis phase to reduce the cost and the time required in performing manual classification.

Results indicate that user requirements classified by our approach are highly consistent with the ones manually classified by the expert (benchmark). Results also show that applying our approach leads to higher accuracy at non-functional requirements as compared to the student's classification accuracy.

However, our approach classification accuracy is slightly lower in classifying the functional requirements compared to the student's classification.

6.2 Future work

Our future work includes evaluating the approach with more and larger case studies and decreasing the human interaction and limitations with our approach by adding extra descriptions and rules to our heuristics. Moreover, it is worth proposing more heuristics for functional requirements in order to reduce incorrect classification. As an extension work of this thesis, it would be very valuable to have an automated classification of Arabic non-functional requirements types such as performance, security, availability, and usability.

Bibliography

[1] Pérez-Verdejo J.M, Sánchez-García A.J, and Ocharán-Hernández J.O. A systematic literature review on machine learning for automated requirements classification. In *2020 8th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pages 21–28. IEEE, 2020.

[2] Wiegers K and Beatty J. *Software requirements*. Pearson Education, 2013.

[3] Gegentana X. A systematic review of automated software engineering. *Master of Science Thesis in Program Software Engineering and Management, University of Gothenburg*, 2011.

[4] Zayed M.A. Automatic software requirements classification: A systematic literature review. *Informatics Bulletin*, pages 29–37, 2021.

[5] IEEE Computer Society. Software Engineering Standards Committee and IEEE-SA Standards Board. *IEEE recommended practice for software requirements specifications*. IEEE, 1998.

[6] Sommerville I. *Software Engineering 9th*. Addison-Wesley, 2011.

[7] Khoufi N, Aloulou C, and Belguith L.H. Parsing arabic using induced probabilistic context free grammar. *International Journal of Speech Technology*, pages 313–323, 2016.

[8] Shehadeh K, Arman N, and Khamayseh F. Semi-automated classification of arabic user requirements into functional and non-functional requirements using nlp tools. In *2021 International Conference on Information Technology (ICIT)*, pages 527–532. IEEE, 2021.

[9] Chakraborty A, Baowaly M.K, Arefin A, and Bahar A.N. The role of requirement engineering in software development life cycle. *Journal of emerging trends in computing and information sciences*, pages 723–729, 2012.

[10] Laplante P.A. *Requirements engineering for software and systems*. Auerbach Publications, 2017.

[11] Windle D.R and Abreo L.R. *Software requirements using the unified process: a practical approach*. Prentice Hall Professional, 2003.

[12] Kar P and Bailey M. Requirements management working group: characteristics of good requirements. In *INCOSE International Symposium*, pages 1225–1233. Wiley Online Library, 1996.

[13] Elazhary H. Avoiding ambiguities in arabic software user requirements. *International Journal of Software Engineering and Its Applications*, pages 141–160, 2016.

[14] Manning C.D, Surdeanu M, Bauer J, Finkel J.R, Bethard S, and McClosky D. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[15] Habash N, Rambow O, and Roth R. Mada+ tokan: A toolkit for arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization. In *Proceedings of the 2nd international conference on Arabic language resources and tools (MEDAR), Cairo, Egypt*, page 62, 2009.

[16] Pasha A, Al-Badrashiny M, Diab M, El Kholy A, Eskander R, Habash N, Pooleery M, Rambow O, and Roth R. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Proceedings of the ninth international conference on language resources and evaluation (LREC'14)*, pages 1094–1101, 2014.

[17] Obeid O, Zalmout N, Khalifa S, Taji D, Oudah M, Inoue B, Alhafniand G, Eryani F, Erdmann A, and Habash N. Camel tools: An open source python toolkit for arabic natural language processing. In *Proceedings of the 12th language resources and evaluation conference*, pages 7022–7032, 2020.

[18] Obeid O. Camel tools releases. https://github.com/CAMeL-Lab/camel_tools/releases. Date accessed: 20.11.2021.

[19] columbia. [mada-users] mada+token download. https://lists.cs.columbia.edu/pipermail/mada-users/2016-May/000116.html. Date accessed: 12.9.2021.

[20] Singh P, Singh D, and Sharma A. Rule-based system for automated classification of non-functional requirements from requirement specifications. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 620–626. IEEE, 2016.

[21] Hussain I, Kosseim L, and Ormandjieva O. Using linguistic knowledge to classify non-functional requirements in srs documents. In *International Conference on Application of Natural Language to Information Systems*, pages 287–298. Springer, 2008.

[22] Sharma V.S, Ramnani R.R, and Sengupta S. A framework for identifying and analyzing non-functional requirements from text. In *Proceedings of the 4th international workshop on twin peaks of requirements and architecture*, pages 1–8, 2014.

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

[24] Kurtanović Z and Maalej W. Automatically classifying functional and non-functional requirements using supervised machine learning. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 490–495. IEEE, 2017.

[25] Haque M.A, Rahman M.A, and Siddik M.S. Non-functional requirements classification with feature extraction and machine learning: An empirical study. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pages 1–5. IEEE, 2019.

[26] Younas M, Jawawi D.N, Ghani I, and Shah M.A. Extraction of non-functional requirement using semantic similarity distance. *Neural Computing and Applications*, pages 7383–7397, 2020.

[27] Rahman M.A, Haque M.A, Tawhid M.N, and Siddik M.S. Classifying non-functional requirements using rnn variants for quality software development. In *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation*, pages 25–30, 2019.

[28] Ott D. Automatic requirement categorization of large natural language specifications at mercedes-benz for review improvements. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 50–64. Springer, 2013.

[29] Knauss E and Ott D. (semi-) automatic categorization of natural language requirements. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 39–54. Springer, 2014.

[30] Winkler J and Vogelsang A. Automatic classification of requirements based on convolutional neural networks. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 39–45. IEEE, 2016.

[31] Jabbarin S and Arman N. Constructing use case models from arabic user requirements in a semi-automated approach. In *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, pages 1–4. IEEE, 2014.

[32] Arman N and Jabbarin S. Generating use case models from arabic user requirements in a semiautomated approach using a natural language processing tool. *Journal of Intelligent Systems*, pages 277–286, 2015.

[33] Arman N. 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, 2015.

[34] Nassar I.N and Khamayseh F.T. Constructing activity diagrams from arabic user requirements using natural language processing tool. In *2015 6th International Conference on Information and Communication Systems (ICICS)*, pages 50–54. IEEE, 2015.

[35] Alami N, Arman N, and Khamyseh F. 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)*, pages 309–314. IEEE, 2017.

[36] Alami N, Arman N, and Khamayseh F. Generating sequence diagrams from arabic user requirements using mada+ tokan tool. *Int. Arab J. Inf. Technol.*, pages 65–72, 2020.

[37] Seresht S.M and Ormandjieva O. Automated assistance for use cases elicitation from user requirements text. In *Proceedings of the 11th Workshop on Requirements Engineering (WER 2008)*, pages 128–139, 2008.

[38] Thakur J.S and Gupta A. Automatic generation of sequence diagram from use case specification. In *Proceedings of the 7th India Software Engineering Conference*, pages 1–6, 2014.

[39] Gutiérrez J.J, Nebut C, Escalona M.J, Mejías M, and Ramos I.M. Visualization of use cases through automatically generated activity diagrams. In *International Conference on Model Driven Engineering Languages and Systems*, pages 83–96. Springer, 2008.

[40] Bajwa I.S and Choudhary M.A. From natural language software specifications to uml class models. In *International Conference on Enterprise Information Systems*, pages 224–237. Springer, 2011.

[41] Yue T, Briand L.C, and Labiche Y. Automatically deriving uml sequence diagrams from use cases. *Simula Research Laboratory*, 2010.

[42] Ilieva M.G and Ormandjieva O. Models derived from automatically analyzed textual user requirements. In *Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06)*, pages 13–21. IEEE, 2006.

[43] Chung L, Nixon B.A, Yu E, and Mylopoulos J. *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media, 2012.

[44] Mustafa M.C, Gencol M, Oguz A, and Gedikli A.M. Ners project, software requirement specification. https://senior.ceng.metu.edu.tr/2016/codewhisperers/documents/SRS.pdf. Date accessed: 27.1.2022.

[45] Hossin M and Sulaiman M.N. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, page 1, 2015.

[46] Danis B. Software requirements specification (srs) for the ¡nodes portal toolkit (npt)¿. https://www.researchgate.net/publication/265397639_Software_Requirements_Specification_SRS_for_the_Nodes_Portal_Toolkit_NPT. Date accessed: 25.2.2022.

[47] Bus L and Eccam. Reqview software requirements specification example. https://www.reqview.com/papers/ReqView-Example_Software_Requirements_Specification_SRS_Document.pdf. Date accessed: 17.3.2022.

Appendix A

CAMeL Installation

You can install CAMeL tools API in two ways [18]:

1. Install using pip by the following command:

   `pip install camel-tools`

   # Or run the following if you already have camel_tools installed

   `pip install camel-tools --upgrade`

2. Install from source using the following command:

   # Clone the repo
   `git clone https://github.com/CAMeL-Lab/camel_tools.git`
   `cd camel_tools`
   # Install from source
   `pip install .`

   # or run the following if you already have camel_tools installed
   `pip install --upgrade .`

Next, we need to tell CAMeL Tools to install the data. This will take a couple of minutes to complete. To install the datasets required by CAMeL Tools components run one of the following:
To install all datasets
`camel_data -i all`
Or just the datasets for morphology and MLE disambiguation only
`camel_data -i light`
Or just the default datasets for each component
`camel_data -i defaults`

After installation, you can call CAMeL tools libraries in your Python code and then use the utilities at the time and manner you need.