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

Karmel Shehadeh
*Deanship of Graduate Studies/M*aster's of Informatics Program
*Palestine Polytechnic University*
Hebron, Palestine
kshehadeh@ppu.edu

Nabil Arman
*Department of Computer Science and Information Technology*
*Palestine Polytechnic University*
Hebron, Palestine
narman@ppu.edu

Faisal Khamayseh
*Department of Computer Science and Information Technology*
*Palestine Polytechnic University*
Hebron, Palestine
faisal@ppu.edu

***Abstract*— Functional and non-functional requirements are equally important in software engineering. Both of them are mixed together within the same software requirement document. Usually, they are expressed in natural languages. So, a lot of human effort is required to classify them. Software requirements classification is a challenging task. Requirements classification can help developers to deliver quality software that meets users' expectations completely. In this paper, we present a Semi-Automated classification approach of Arabic functional and non-functional requirements using a natural language processing (NLP) tool. We propose a set of heuristics based on basic constructs of Arabic sentences in order to extract information from Arabic software requirements to classify the requirements into functional and non-functional requirements. This research aims to help software engineers by reducing the cost and time required in performing manual classification of software requirements.**

***Keywords— Requirements Classification, Automated Software Engineering, NLP Tools, Functional Requirements, Non-Functional Requirements.***

## I. INTRODUCTION

The functional requirements (FR) of a system describe what the system should do and Non-Functional Requirements (NFR) 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 [1].

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 functional requirements 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 [2].

There are several techniques for software requirements classification. According to the review of the literature, most of the techniques for software requirements classification are 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.

Manual classification of software requirements from the software specification document is a challenging and time-consuming task. In our research, we attempt to solve this drawback by providing a Semi-Automated classification of software requirements using an NLP tool parser with a set of heuristics based on basic constructs of Arabic sentences. We will present our approach towards an effective technique for semi-automatic classification of textual requirements into two categories, namely FRs and NFRs. The main objective of this research is to propose a novel approach to classify Arabic functional and non-functional requirements using a semi-automated method based on NLP tools.

The rest of the paper is organized as follows: Section II and III highlight the related work and background respectively. Section IV presents the proposed classification approach. Section V is the section of validation. Finally, the conclusion is presented in section VI.

## II. RELATED WORK

### *A. Software Requirements Classification Researches*

Hussain et al [3] presented 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, and extracts the values of all the features chosen by the authors, and uses Weka to train C4.5 decision tree learning algorithms.

Singh. et al in this paper [4] 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 1 [4], 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.

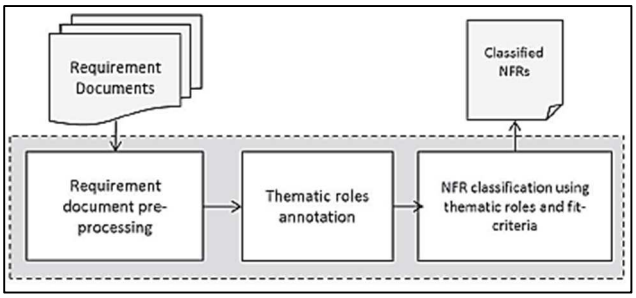


Figure 1. Three-phase system design.

Kurtanovi´c and Maalej [5] 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 [6] proposed an automatic NFR classification approach for quality 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, SVM, 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 input of machine learning algorithms. The whole process of this framework is divided into four steps as shown in Figure 2 [6] which include: Data Preprocessing, Feature Extraction, Train Classifier, 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.

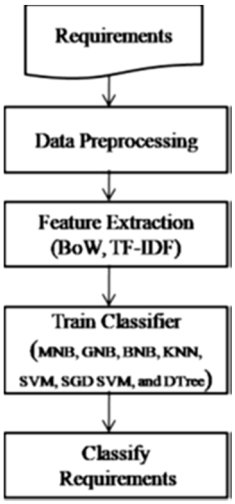


Figure 2. Proposed Method Overview.

Younas et al [7] 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.

*B. Automated Software Arabic Requirement Researches using NLP tools:*

Jabbarin and Arman [8][9] 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 [10] 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 [11] proposed a semi-automated approach for constructing 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 [12][13] 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.

## III. BACKGROUND

Arabic language is a prominent member of the Semitic languages 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 well informed sentences.

Arabic language is not an easy task to parse because first, the particularities of the Arabic language make it more ambiguous than other natural languages this is due to its morphological [14], syntactic and semantic characteristics, second the significant lack of digital resources of the Arabic language, especially concerning the grammars and corpora [15].

### *A. Software Requirements Specification*

The software requirements specification (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 [2].

The level of detail required 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. Also, 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 [2].

### *B. Arabic User Requirements*

User and system requirements are usually written in natural languages supplemented by relevant diagrams and tables. It should be clear, unambiguous, complete, easy to understand, and consistent [2].

### *C. Natural Language Processing Tools*

There are many tools for Arabic morphological analysis. Each has different Characteristics. This is based on how those tools are developed and what database is used. There are four tools that are freely available and very suitable for tokenizing Arabic text: Stanford, MADA+TOKAN, CAMEL POS and MADAMIRA Tools:

- 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, sentence splitting [16].
- 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 [17].
- 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 [18].
- 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 [19].

## IV. PROPOSED CLASSIFICATION APPROACH

We propose a high-level approach to illustrate the detailed steps to classify Arabic user requirements into functional and non-functional requirements. To analyze Arabic user requirements, we present the basic grammar rules for the simple verbal sentences [20] as the following:

- Sentence → Nominal Sentence | Verbal Sentence.
- Nominal Sentence:
  1. Nominal Sentence → [Annuler] + Subject Phrase + Predicate Phrase.
  2. Annuler → Original Particle | Transformed Particle.
  3. Original Particle → Negative Particle | Interrogative Particle | Preposition.
  4. Transformed Particle → Verbal Transformed Particle | Adjective Preposition.
  5. Adjective Particle → “Inna” and Her Sister.
  6. Subject Phrase → Subject + [Expansion Phrase].
  7. Subject → Noun | Pronoun.
  8. Adverb → Adverb of Place | Adverb of Time.
- Verbal Sentence:
  1. Verbal Sentence → Verb + subject + object.
  2. Verb → Past | Present | Imperative mood.
  3. Subject → Noun | Pronoun.
  4. Noun → Proper noun | demonstrative |Relative.
  5. Object → Noun | Quasi Sentence.
  6. Quasi Sentence → Preposition phrase |Adverb phrase.
  7. Preposition Phrase → Preposition + Noun.
  8. Adverb Phrase → Adverb + Noun.

To analyze the Arabic sentence, we will use an NLP tool for parsing, tokenization, part of speech, and sentence splitting.

Currently, we are reviewing several software graduation projects for PPU students and SRS documents for developers to extract features that distinguish functional from non-functional requirements. These features will be used in proposing a set of heuristics for our approach. We extracted the following features that distinguish non-functional from functional requirements:

1. The appearance of cardinals indicates high probability of NFR.
2. The appearance of non-Arabic words often denotes techniques, like (SQL, HTML…etc.). They are usually found in the description of NFR.
3. The appearance of adjectives/adverbs indicates a high probability of NFR.
4. The sentence contains keywords of non-functional requirements, that have been mentioned in [20].

Finally, we will develop a Python-based classification program to evaluate our proposed heuristics. Figure 3. illustrates our approach as a block diagram.

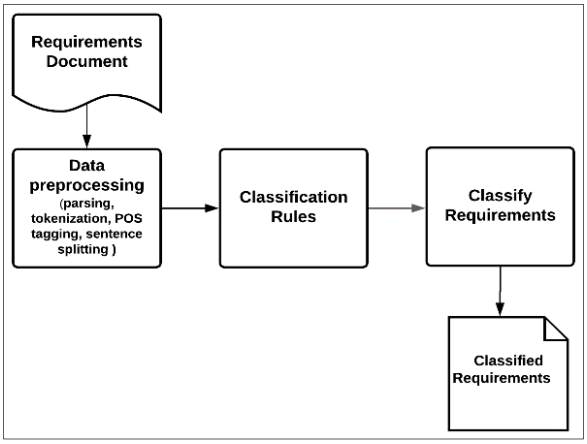


Figure 3. Proposed Approach Block Diagram.

For the advantage of loose coupling, so that the program could be flexible, testable and maintainable, we apply the dependency injection approach in our research. We break down our logic into different classes, each one is responsible for a single task, then these classes can collaborate to complete the job. For example, a class for analyzing the tags based on the proposed heuristics to classify the requirements will depend on the services of the class that tags the tokens of the statement.

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

1. The appearance of cardinals/numbers is indicating a high probability of NFR.
   If a cardinals\numbers are existing in sentences in any sentence syntax then it raises the likelihood that the sentence is a non-functional requirement such as the following sentences:
   a) For example:" يقوم النظام بتحديث العرض كل ستين ثانية"
      A translation of the example:" The system updates the display every sixty seconds"
      Using MADAMIRA POS, the statement is divided into: يقوم/Verb النظام/Noun بتحديث/Noun ستين/Number Quantity Noun/ كل/Noun العرض/Noun ثانية/Numerical Adjectives
   b) For example:" يجب أن يكون المشاهدين قادرين على قراءة بيانات الحدث من مسافة عرض تبلغ 30 متر"
      A translation of the example:" Viewers should be able to read event data from a viewing distance of 30 meters"
      Using MADAMIRA POS, the statement is divided into: يجب/Verb أن/Subordinating Conjunction يكون/Verb المشاهدين/Noun قادرين/Adjective على/ Preposition قراءة/Noun بيانات/noun الحدث/Noun من/ Preposition مسافة/Noun عرض/Noun تبلغ/Verb 30/Digit

   As shown in the above examples, POS of cardinals/numbers using MADAMIRA is "Number Noun" if it is written in letters and "Digit" if it is written in numbers.

2. The appearance of non-Arabic words often denotes techniques, 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 such as the following sentences:
   a) For example:" Windows XP يعمل النظام ضمن نظام التشغيل"
      A translation of the example:" The system works within the Windows XP operating system"
      Using MADAMIRA POS, the statement is divided into: يعمل/Verb النظام/Noun ضمن/Noun نظام/Noun التشغيل/Noun Windows/Foreign Xp/Foreign
   b) For example:" MySQL أو HSQL سيحتاج نظام إدارة قواعد البيانات "
      A translation of the example:" The database management system will need MySQL or HSQL"
      Using MADAMIRA POS, the statement is divided into: يحتاج/Verb نظام/Noun ادارة/Noun قواعد/Noun البيانات/Noun MySQL/Foreign أو/Conjunction HSQL/Foreign

   As shown in the above examples, POS of non-Arabic words using MADAMIRA is "Foreign".

3. The appearance of adjectives/adverbs indicates a high probability of NFR. We propose a set of heuristics for verbal and nominal sentences as follows:
   a) For example:" ًيجب أن يكون النظام بديهياً و واضحا"
      A translation of the example:" The product should be intuitive and clear"

Using MADAMIRA POS, the statement is divided into: يجب/Verb أن/Subordinating conjunction يكون/Verb النظام/Noun و/Conjunction بديهياً/Adjective واضحاً/Adjective

b) For example:" النظام يجب أن يكون بديهياً وواضحاً"
A translation of the example:" The product should be intuitive and clear"
Using MADAMIRA POS, the statement is divided into: النظام/Noun يجب/Verb أن/Subordinating conjunction يكون/Verb و/Conjunction بديهياً/Adjective واضحاً/Adjective
As shown in the above examples, POS of the adjectives using MADAMIRA is "Adjective" and the adverbs is "Adverb".

4. If the sentence contains keywords of non-functional requirements, that have been mentioned in [20]. Then it raises the likelihood that the sentence is a non-functional requirement such as the following sentences:

We propose a set of heuristics for the nominal sentences as follows:

a) If the statement is simple:
For example:" سهولة التعامل مع النظام"
A translation of the example:" Ease of handling the system"
Using MADAMIRA POS, the statement is divided into: سهولة/Noun التعامل /Noun مع/Noun النظام/Noun
The main subject is سهولة.

b) If the statement contains the connector (و) without any verb or subject in the second statement:
For example:" أمان النظام وسرية البيانات"
A translation of the example:" System security and data confidentiality"
Using MADAMIRA POS, the statement is divided into: أمان/Noun النظام/Noun و/Conjunction البيانات/Noun
The first subject is أمان.
The second subject is سرية.

We propose a set of heuristics for the verbal sentences. These heuristics presented as follows:

a) If the statement is simple:
For example:" يقدم النظام مرونة عالية لمستخدميه"
A translation of the example:" The system offers high flexibility to its users"
Using MADAMIRA POS, the statement is divided into: يقدم/Verb النظام /Noun مرونة/Adjective عالية/Noun لمستخدميه/Noun
The main object is مرونة.

b) If the statement contains the connector (و) without any verb or actor in the second statement:
For example," يقدم النظام مرونة عالية و سهولة لمستخدميه"
A translation of the example: "The system offers high flexibility and ease to its users."
Using MADAMIRA POS, the statement is divided into: يقدم/Verb النظام /Noun مرونة/Adjective عالية/Noun و/Conjunction سهولة/Noun لمستخدميه/Noun
The first object is مرونة.
The second object is سهولة.

We can determine whether this sentence is a non-functional requirement by returning the subjects/objects to their Lemma using MADAMIRA tool Lemmas then comparing them with the lemma of Keywords [21].

## V. EVALUATION

The next step in this research is the evaluation of the proposed approach. The purpose of this evaluation is to calculate the accuracy of our approach and to compare the result with previous related researches and to check if our approach is better than these researches or not. To evaluate the accuracy of our approach, we will perform a test of one-third of the pre-classified requirements that we will use for this study. Then we will measure two evaluation metrics: recall and precision. These metrics are used to evaluate natural language processing-based knowledge extraction systems as of the following [22]:

*a)* Recall: The completeness of the results produced by the system.

*b)* Precision: It expresses the accuracy of the designed system.

## VI. CONCLUSION

In this paper, we proposed a novel approach for classification of Arabic user requirements into functional and nun functional requirement. 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. This research aims to help software engineers in the analysis phase by reducing the cost and the time required in performing manual classification.

## REFERENCES


[1] IEEE COMPUTER SOCIETY. SOFTWARE ENGINEERING STANDARDS COMMITTEE AND IEEE-SA STANDARDS BOARD, 1998. *IEEE RECOMMENDED PRACTICE FOR SOFTWARE REQUIREMENTS SPECIFICATIONS* (VOL. 830, NO. 1998). IEEE.

[2] Sommerville, I., 2011. Software engineering 9th Edition. *ISBN-10*, *137035152*, p.18

[3] Hussain, I., Kosseim, L. and Ormandjieva, O., 2008, June. Using linguistic knowledge to classify non-functional requirements in SRS documents. In *International Conference on Application of Natural Language to Information Systems* (pp. 287-298). Springer, Berlin, Heidelberg.

[4] Singh, P., Singh, D. and Sharma, A., 2016, September. 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)* (pp. 620-626). IEEE.

[5] Kurtanović, Z. and Maalej, W., 2017, September. Automatically classification functional and non-functional requirements using supervised machine learning. In *2017 IEEE 25th International Requirements Engineering Conference (RE)* (pp. 490-495). Ieee.

[6] Haque, M.A., Rahman, M.A. and Siddik, M.S., 2019, May. 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)* (pp. 1-5). IEEE.

[7] Younas, M., Jawawi, D.N., Ghani, I. and Shah, M.A., 2020. Extraction of non-functional requirement using semantic similarity distance. *Neural Computing and Applications*, *32*(11), pp.7383-7397.

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

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

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

[11] Nassar, I.N. and Khamayseh, F.T., 2015, April. Constructing activity diagrams from Arabic user requirements using Natural Language Processing tool. In 2015 6th International Conference on Information and Communication Systems (ICICS) (pp. 50-54). IEEE.

[12] Alami, N., Arman, N. and Khamyseh, F., 2017, May. A semi-automated approach for generating sequence diagrams from Arabic user requirements using a natural language processing tool. In 2017 8th International Conference on Information Technology (ICIT) (pp. 309-314). IEEE.

[13] Alami, N., Arman, N. and Khamayseh, F., 2020. Generating sequence diagrams from arabic user requirements using mada+ tokan tool. Int. Arab J. Inf. Technol., 17(1), pp.65-72.

[14] Alqrainy, S., Muaidi, H., & Alkoffash, M. S. (2012). Context-free grammar analysis for Arabic sentences. International Journal of Computer Applications, 53(3).

[15] Khoufi, N., Aloulou, C., & Belguith, L. H. (2016). Parsing Arabic using induced probabilistic context free grammar. International Journal of Speech Technology, 19(2), 313-323.

[16] Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S. and McClosky, D., 2014, June. The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60).

[17] Habash, N., Rambow, O. and Roth, R., 2009, April. 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 (Vol. 41, p. 62).

[18] Obeid, O., Zalmout, N., Khalifa, S., Taji, D., Oudah, M., Alhafni, B., Inoue, G., Eryani, F., Erdmann, A. and Habash, N., 2020, May. CAMeL tools: An open source python toolkit for Arabic natural language processing. In Proceedings of the 12th language resources and evaluation conference (pp. 7022-7032).

[19] Pasha, A., Al-Badrashiny, M., Diab, M.T., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O. and Roth, R., 2014, May. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In LREC (Vol. 14, No. 2014, pp. 1094-1101).

[20] Daimi, K., 2001. Identifying syntactic ambiguities in single-parse Arabic sentence. *Computers and the Humanities*, *35*(3), pp.333-349.

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

[22] Bajwa, I. S., and Choudhary, M. A. (2012). From natural language software specifications to UML class models. In Enterprise Information Systems (pp. 224-237). Springer Berlin Heidelberg.