IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 31, No. 6, June, 2026, pp. 1903-1913
Journal homepage: http://iieta.org/journals/isi

A Semi-Automated Approach for Deriving UML Use Case Elements from Functional Requirements

Anfal A. Fadhil*, Asmaa Hadi Al Bayati

Software Department, College of Computer Science and Mathematics, University of Mosul, Mosul 41001, Iraq

Corresponding Author Email: anfalaaf@uomosul.edu.iq

https://doi.org/10.18280/isi.310613

Received: 9 February 2026
Revised: 8 April 2026
Accepted: 20 April 2026
Available online: 30 June 2026

Keywords:
software requirements, Unified Modeling Language use case modeling, natural language processing, Random Forest, requirements engineering, automated model generation

ABSTRACT

Software requirements are commonly documented in natural language, transforming functional requirements into formal software models a time-consuming and error-prone task. Although Unified Modeling Language (UML) use case diagrams are widely adopted for representing system functionality, their manual construction requires considerable effort from requirements analysts and is affected by ambiguity and inconsistency in textual specifications. This study proposes a semi-automated framework for extracting UML use case elements from natural language software requirements by integrating natural language processing (NLP) techniques with a Random Forest (RF)-based classification approach. The proposed framework performs text preprocessing, linguistic feature extraction, actor and use case identification, and irrelevant element filtering to generate preliminary UML use case components. The RF classifier is employed to distinguish meaningful elements from redundant or unrelated candidates generated during the NLP stage. The proposed approach was evaluated using functional requirement datasets collected from multiple software systems, including E-Store, Inventory, Philips, and Mental Health Care - Patient Management System (MHC-PMS). Experimental results demonstrate that the framework achieved an average precision of 73.5% and recall of 71.5%, showing improved performance compared with conventional NLP-based extraction methods. The proposed approach reduces manual effort in requirements analysis and provides practical support for early-stage UML modeling. However, further validation with larger and more diverse requirement datasets is required to improve generalizability.

1. INTRODUCTION

In systems engineering, the requirements analysis phase constitutes the most critical stage of the development process. The primary objective of this phase is to define requirements documents with precision and clarity. Such requirements are typically expressed in natural language; however, natural language descriptions are often ambiguous, unstructured, and prone to misinterpretation. It is therefore necessary to analyze natural-language requirements and translate them into a formal specification, such as Unified Modeling Language (UML). Object-Oriented Analysis and Design (OOAD), which represents system domains using UML diagrams, plays an essential role in modern software development by bridging the gap between informal requirements and rigorous, implementable system models [1, 2].

The task of converting user requirements into UML diagrams falls to human analysts. Money, time, and effort are all heavily invested in this process. Furthermore, inconsistency and ambiguity are examples of natural language problems that can lead to errors. Because of all of this, semi-automated or automated support is required [3, 4].

Among the methods for describing the functional requirements is the UML use case diagram. The information is extracted using Natural Language Processing (NLP) since the requirements have been written in natural language. Text information extraction was accomplished using NLP [5, 6].

NLP is typically utilized for transforming extracted data to a machine-understandable format. However, ambiguous situations and different NLP approaches result from the grammar rules and various natural language structures [7, 8]. These days, NLP is widely employed in a variety of applications that take human-computer interaction (HCI) into account. One of the most important tasks in the software lifecycle phases is creating UML use case diagrams as part of the software analysis. Understanding system requirements is aided by the creation of use case diagrams. Yet, because system requirements need investigation, it takes time to create use case diagrams [9]. The UML's use case diagrams show how the system communicates with outside parties (actors) to accomplish a task. The external entity interacting with the system is known as the actor. It could be a system, user, device, company, and so on [10, 11]. Apart from the actor, the use case diagram illustrates the capabilities of the system through the use cases and their relations. Despite efforts, making a use case diagram with NLP remains a difficult task.

The Random Forest (RF) algorithm, a machine learning (ML) approach, and NLP are used in the presented study to suggest a novel method for creating a UML use case diagram. While NLP examines the functional requirements, the

designed solution expedites the process by performing early spelling checks. The system recognizes parts of speech (POS) and assigns particular grammatical patterns to English sentences through segmenting them. After that, it first determines the use case and the actor. The RF algorithm is used for eliminating unwanted actors, duplicates, and use cases. The use case diagram is then drawn. Our study's primary contributions are:

1. We have developed an algorithm for extracting actors, relationships, and use cases from the functional requirements. Both external and primary actors are extracted by our method.
2. Our suggested method is validated by using literature-based case investigations, which demonstrate its importance in contrast to the novel method.
3. It employs NLP and the RF algorithm for element extraction as well as graph visualization.
4. For evaluating outcomes from literature-based case studies, we utilized recall and precision. Our method produces significant outcomes.

The remainder of the present study is structured as follows: Section 2 presents the literature that is reviewed in related work. Section 3 describes the software requirements, NLP, and the RF algorithm. Section 4 explains how the suggested method is used. Section 5 presents the case study's specifics, findings, and a comparison with an existing method. Finally, the study's conclusion comes last in Section 6.

2. RELATED WORK

The creation of use case diagrams has been the subject of extensive research. As a result, several papers have emerged analyzing the latest efforts in the main field (UML diagrams) and the field itself (use case diagrams) between 2018 and 2024. Table 1 provides a summary of these efforts.

Table 1. An overview of previous studies

Year	Source	Diagrams	Classification	Method	Mechanism to Generate UML Diagram (Automated or Manual)
2018	[12]	Use case	Rule-based approaches	NLP techniques, tree tagger parser	Automated
2020	[13]	Class diagram	Rule-based approaches	NLP techniques, heuristic rules	Manual
2021	[14]	Sequence Diagrams	Rule-based approaches	NLP techniques, rule-based decision approaches	Automated
2018	[15]	Use case	NLP-based UML generation	SRL, NLP techniques	Manual
2019	[16]	Use case	NLP-based UML generation	Knowledge process of acquisition based upon the verbal semantics	--
2021	[6]	Use case Activity	NLP-based UML generation	NLP techniques, heuristic rules	Automated
2023	[17]	Use case	NLP-based UML generation	NLP techniques, network science	Automated
2023	[18]	Use case	Machine learning methods	NLP techniques, Naive Bayes algorithm	Manual
2024	[19]	Sequence Diagrams	Recent LLM-based approaches	ChatGPT	Automated

Note: UML = Unified Modeling Language; NLP = Natural Language Processing; SRL= Semantic Role Labeling.

All these related works depend on NLP except the last and can be classified in many specific areas as:

2.1 Rule-based approaches

In 2018, Elallaoui et al. [12] proposed a method for generating relationships, actors, and use cases. Only user stories generated using the Wautelet template are used to create these. The method identifies actors as well as use cases with the use of a few predefined nouns, verbs, and POS tags. They use a single case study to assess their method. The method is innovative in bridging user stories and UML use cases, but its disadvantages include rigid reliance on user story format and limited NLP coverage.

In order to create UML models, researchers [13] concentrated on the automation of the procedure of obtaining useful data from Natural Language (NL) texts. The ambiguity regarding NL causes the current methods to be less accurate. NL practices, as well as a set of heuristic rules in order to aid in transformation, are used in this research to offer an approach for generating class models from the software design requirements. To improve the resulting class diagram's accuracy, the requirements of the NL are transformed into a controlled, formal representation. To create a UML class diagram from specifications of provided requirements, a set of predefined rules was established for extracting OO concepts, including attributes, classes, relationships, and methods. Practical applications and evaluations of the approach have demonstrated its viability and acceptability. The method depends heavily on handcrafted heuristic rules for mapping text elements to UML concepts.

The study [14] offered an automated method for turning the textual use cases that have been written in natural language into behavioral models that are represented as UML sequence diagrams. The method identifies interactions and objects at the problem level by combining several NLP methods with some rule-based decision techniques. Furthermore, various quality criteria are established to evaluate the generated sequence diagrams' validity in relation to the expected behavior of a specific use case. Comparable experiments could be evaluated using criteria that have been developed in this research to assess the quality of the analysis sequence diagrams. With the use of several case studies, we assess our approach in terms of the accuracy and comprehensiveness of the sequence diagrams produced by employing those measures. The approach works best on well-structured sentences and may struggle with pronouns, implicit actors, or domain-specific terminology.

2.2 Natural Language Processing-based Unified Modeling Language generation

With the use of semantic role labeling methods, Jebril et al. [14] suggest a semi-automated method for identifying actors as well as use cases from requirements.

The paper states the approach was manually tested utilizing known instances and shown its validity, but it does not report a systematic dataset or quantitative metrics (precision/recall/F1).

In 2019, Vasques et al. [15] proposed a method for offering relevant knowledge for the extraction of the use cases. This method addresses many issues, such as inconsistency, incompleteness, and redundancy. The verbal method has been employed. The analyst prepared data for this method. Subject-action-object formation, converting passive voice to the active form, and splitting complicated sentences into smaller ones are all examples of text preparation. The verbs are then chosen for thematic proto-role assignment as well as syntax-based classification. After that, related concepts are clustered. The purpose of the concept map is to identify cause-and-effect relations. The concept map has the form of a network that has edges of various colors. A specific correlation is represented by each color. Semantically structured statements linked to a certain actor are also included in this study, along with potential use cases. A UML diagram is produced using such actions, actors, and use cases. A key limitation of this research is that it was validated on only one or a few illustrative examples, providing limited evidence of generalizability to other domains, larger requirement sets, or noisy real-world requirements.

This makes it brittle, domain-dependent, and difficult to generalize to diverse requirement styles or new domains.

In 2021, Abdelnabi and Maatuk [6] suggested an approach for making use cases and activity diagrams based on the requirements in natural language. Activity diagrams and use cases are produced by NLP approaches and established rules. Among such NLP approaches are stemming, tokenization, typed dependency, lemmatization, and grammatical relation recognition. For the purpose of generating UML diagrams, textual requirements must be written or normalized with the use of sixteen specified rules. We are also encouraged to utilize OpenIE CoreNLP for the generation of triplets and to exclude the word "system" from an actor. Nevertheless, our method has produced precise triplets using OpenIE 5.0 with OpenIE CoreNLP. Along with NLP approaches, our study has made use of network science and semantic similarity. We create use cases using both the predicates and the objects. Furthermore, our method for identifying subject tags does not rely on type dependency. Only OpenIE has been utilized for triplet generation. The approach is mainly built on heuristic rules and NLP parsing, and the system does not robustly address issues like synonyms or multi-sentence flows, which reduces accuracy.

The problem of automated artifact extraction from the NL requirements is solved in the study by Malik et al. [16]. They have provided an automated method for the use of the natural language requirements to make actors, utilize cases, and their relationships. Neither formalism nor human intervention is used in our suggested method. We have utilized network science and NLP to automate the suggested method. The results of our suggested method for extracting use case elements from the NL requirements are encouraging. We use a number of case examples from the literature to validate the suggested approach. The importance of the suggested method for extracting use case elements from NL requirements is demonstrated by evaluating the method on the literature-based case studies. The method lowers the amount of human effort required for designing and developing software. Combining NLP with network science adds algorithmic overhead.

2.3 Machine learning methods

The study by Dias et al. [17] offered a novel method for creating a use case diagram through the combination of ML and NLP to analyze the provided user story. During the SDLC's design phase, use case diagrams are crucial. This demonstrates how much effort and time could be saved by automating the use case diagram design process. Many semi-automated and manual tools have already been created. The necessity of use case diagrams and the challenges that have been encountered in their design are covered in this work. Through generating the use case diagram, this work aims to address those problems. The paper does not provide detailed precision, recall, or F1 metrics across multiple datasets.

2.4 Recent large language model-based approaches

The study by Ferrari et al. [18] examined ChatGPT's capability to produce a particular type of model—UML sequence diagrams—from NL specifications. The sequence diagrams produced through ChatGPT for 28 requirements papers of different types and from different domains are examined in this qualitative study. Through evaluation logs, observations from the study of the created diagrams were systematically captured, and thematic analysis was used to categorize them. Our findings show that while the models generally meet the standard and are reasonably comprehensible, there are frequently issues with their accuracy and completeness in relation to the requirements. This problem is most noticeable when requirements are inconsistent and ambiguity is present. The research's conclusions have the potential to impact the real-world application of LLMs in the RE process and pave the way for innovative RE-specific prompting techniques aimed at efficient model development. The study uses 28 requirement documents across domains, which is moderate but still limited.

3. TECHNICAL BACKGROUND AND PRELIMINARIES

Before beginning the research methodology, there is a need to understand the technical background necessary to implement the proposed approach. These techniques include:

3.1 Create software requirements

The descriptions of what a system must accomplish, the services it offers, and limitations on how it can operate are called requirements. Customers' needs for a system that fulfills a specific function, such as placing an order, controlling a device, or locating information, are reflected in such criteria. The procedure for analyzing, finding out, checking, and documenting such limitations and services. In the software industry, the term "requirement" isn't always utilized consistently [20, 21].

Sometimes a requirement is just a limitation on a system or

a high-level, abstract specification of a service that must be offered by a system. It is a thorough, formal definition regarding a system function at the other extreme. Since various readers use requirements differently, they must be read by managers who are not interested in the system's specific features and aren't worried about how the system will be implemented. Since they're involved in system implementation or are interested in the way that the system will support business activities, system requirement readers need to know more specific information about what will be performed by the system [22].

Non-functional and functional requirements are two common categories for software system requirements:

1- Functional requirements, which represent statements of the services that a system should provide [23].

2- Non-functional requirements, which include constraints on functions or services that are offered by the system, including constraints on timing and constraints on the process of development, in addition to the constraints that are imposed by the standards. Non-functional requirements [24, 25].

The use case diagrams in these papers were built based on functional needs. They were extracted from the systems' Software Requirements Specifications (SRS) documents in order to meet functional requirements.

3.2 Natural Language Processing

A field in AI and ML called NLP enables devices to comprehend and work with natural human language. Additionally, to eliminate the need for human processing, natural language requirement texts are analyzed and processed using NLP methods [26]. After receiving the functional requirement text, the suggested study applies various NLP methods to the input text, which include multiple processes:

1. Spelling and grammar correction: It may be that the given functional requirement contains grammatical errors; it will result in predictions that are not accurate. So, one of the vital tasks is grammatical error correction in the area [27]. This suggested method will be applied. The Gingerit library will perform the detection as well as the correction of typing errors in text. This step involves taking the functional requirement as input, fixing any grammar errors, and then passing the grammatically accurate functional requirement to the NLP model.

2. Tokenization: With the use of sentence segmentation, separate into sentences (functional requirement). The sentences will next be divided into individual words with the use of word tokenization. For instance, following tokenization, "I love to cook food" will become [love], [I], [cook], [to], [food]. These techniques aid in carefully going over every word [28].

3. POS tagging is carried out. POS offers useful data regarding tokens or words [14]. To determine if a word is a verb, numeral, noun, preposition, pronoun, adjective, article, interjection, or conjunction, POS tagging is employed [29, 30].

4. Dependency parsing: Every word in the sentence will be subjected to dependency parsing, a method for examining the grammatical relationships between words. The sentence's tokens, or words, might be broken down according to the grammatical relation to produce a tree or graph data structure. The use of dependency parsing results in a precise NLP model. When a sentence does not have a "PRON" (pronoun) in POS tagging, the algorithms could use the "nsub" dependency parser to determine the sentence's subject. After that, it might determine the sentence's verb and object. The dependency parser regarding "dobj" will be used for identifying the object, and POS tagging of "VERB" will be used for identifying the verb [31].

5. Lemmatization: Lemmatization will be done to the subject, verb, and object after they have been identified. As indicated in Table 2, lemmatization can be defined as the procedure of converting a word or token to its root word [32, 33].

6. Identify actors and use cases: the verb-object combination will be recognized as the use case, and distinct subjects will be recognized as actors.

7. Determine the relationship of association between the use cases and actors. This relationship will be determined as follows: following the identification of a new actor, all subsequent use cases are regarded as those of that actor until the emergence of a different and new actor. A dictionary data structure will hold the use case diagram's extracted data.

Table 2. Word and its lemmatization

Word	Lemmatization
Registers	register
Is	be
Regarded	regard

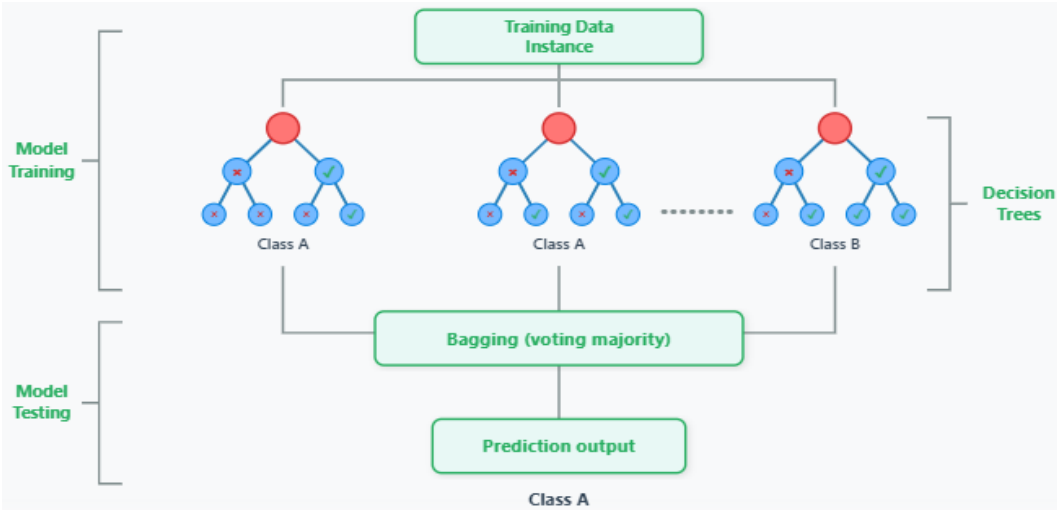

Figure 1. The Random Forest (RF) algorithm in machine learning

3.3 Random Forest algorithm

One ML model employed for classification and forecasting is the RF. A considerable amount of high-quality data is highly important for the efficient collection of data to train AI models and ML algorithms. System performance data is necessary for improving hardware and software efficiency, assessing user behavior, optimizing algorithms, and facilitating pattern recognition, predictive modeling, decision-making, and problem-solving [34, 35].

Several decision trees are generated to build RFs. This is accomplished by selecting input features at random and collecting random data samples with the use of bootstrap samples. Every decision tree is regarded as being simple [36]. Instead of using the original sample, bootstrap samples are used to build each tree. This lessens overfitting and is referred to as bootstrap aggregating, or just bagging [37].

Individual decision trees are simply interpretable, yet such interpretability is lost in RFs due to the accumulation of multiple decision trees (DTs). RFs, on the other hand, frequently outperform other prediction methods. In comparison with DTs, the error rate is more precisely calculated using the RF algorithm. More specifically, it has been mathematically proven that the error rate always converges as the number of trees increases [38].

The high accuracy of RFs in comparison to other methods like boosting and bagging is one of their advantages. Additionally, they work well with large databases and have a large variable capacity, which enables us to examine thousands of input variables without removing any of them. An unbiased estimation of the configuration error is required for balancing the category error in unbalanced data sets and evaluating the significance of variables; these benefits show the effectiveness and power of RFs. Because randomization works well in several real-world applications, it is an often-utilized method in ML and DM [39, 40].

The process starts with a dataset that contains rows and the class labels (columns) that correlate to those rows (Figure 1).

• After that, the training data is used to generate several DTs. A random subset of features and a random subset of data (with replacement) are used for training each tree. This method is referred to as bootstrap aggregating or bagging.

• The ensemble's DTs each develop their own prediction skills.

• Every DT in the ensemble produces a prediction in response to an unseen, new instance.

All of the DTs' predictions are combined to get the final prediction. Usually, a majority vote is used for classification, while averaging is used for regression.

The natural work of RF is that whenever the input space is partitioned into subsets represented in B, draw a bootstrap sample, randomly select a subset, and predict optimally, as shown in Algorithm 1.

A dictionary data structure containing use cases, actors, and association relationships is obtained from the NLP model using the RF, which is the technique suggested in these papers. To improve the filtering, selection, and elimination of unnecessary actors, use cases, and relationships from the data structure in the NLP model, the RF algorithm will be employed. The developed methodology was applied to a set of databases containing 1300 functional requirements (statements in the English language) taken from several systems [41] (such as THEMAS, The Energy Management System, E-GOVERNANCE MISSION MODE PROJECT, and Clarus Weather System). The RF algorithm was trained on 80% of this data and tested on 20% of it. RF is a supervised classification technique that performs classification through a straightforward yet efficient probabilistic approach. The modified data structure will be sent to the model of diagram generation once the actor, use cases, and relationships that are no longer needed have been filtered out.

Algorithm 1. Pseudo-code for Random Forest algorithm using Bootstrap samples

1: start
2: for $i \rightarrow 1$ to B
3: Draw a bootstrap sample of size N from training data
4: while node size ≠ minimum node size do
5: random selection of a subset of m predictor variables from total p
6: for $j = 1$ to m do
7: if j-th predictor optimizes the criterion of splitting then
8: split internal node to 2 child nodes
9: break
10: endFor
11: endWhile
12: endFor
13: end
14: return ensemble tree of all B sub-trees that have been generated in outer for loop

4. METHODOLOGY

The methodology of this paper describes an innovative tool for the generation of use case diagrams from the software requirements, as seen in Figure 2. The methodology starts with reading functional SRS documents written in natural language (English). Then, researchers applied the role of NLP, which contains a set of processes in the statements, such as: Text Cleaning and sentence correction, tokenization (The sentences will next be split into individual words), POS Tagging, identifying nouns that represent candidate actors (e.g., customer, system), identifying verbs that represent candidate use cases (e.g., log in), and finally, lemmatization. The output of NLP for each sentence extracts features such as the presence of modal verbs (shall, must, which represent requirement indicators). Sentence length and keyword frequency. POS pattern (e.g., NOUN-VERB-NOUN, which is equivalent to strong actor-action-object). This will determine all features (actor, use case, and relationship).

When we extract candidate actors and use cases from functional requirements utilizing NLP, we often get:

Synonyms (e.g., Customer vs. Client).

Generic terms (system, application).

Duplicates (Place order vs. Submit order).

Irrelevant phrases (the system shall be fast- that means fast is not a use case). The solution to these problems requires a classifier to decide: Keep or Discard. It has been using RF for Filtering. The RF is an ensemble of many decision trees. It helps reduce overfitting and improves accuracy. It contains feature engineering (input to the classifier) for each candidate actor/use case, computing features such as:

•TF-IDF score: TF (Term Frequency: How often a word appears in a document). IDF (Inverse Document Frequency: How rare a word is across all documents. TF-IDF score combines these to give high weight to important words that appear often in a document.

•POS pattern: actor (noun), use case (verb phrase).
•Word length/frequency.
•Semantic similarity helps cluster them together around domain concepts.
•Requirement position: terms in the first sentences are often important.

The input of the training phase in the RF is candidate features, and the output is a label (1 = useful actor/use case, 0 = noise/duplicate). The RF builds N decision trees on different subsets of data, adding features. The final classification is the majority voting of all trees.

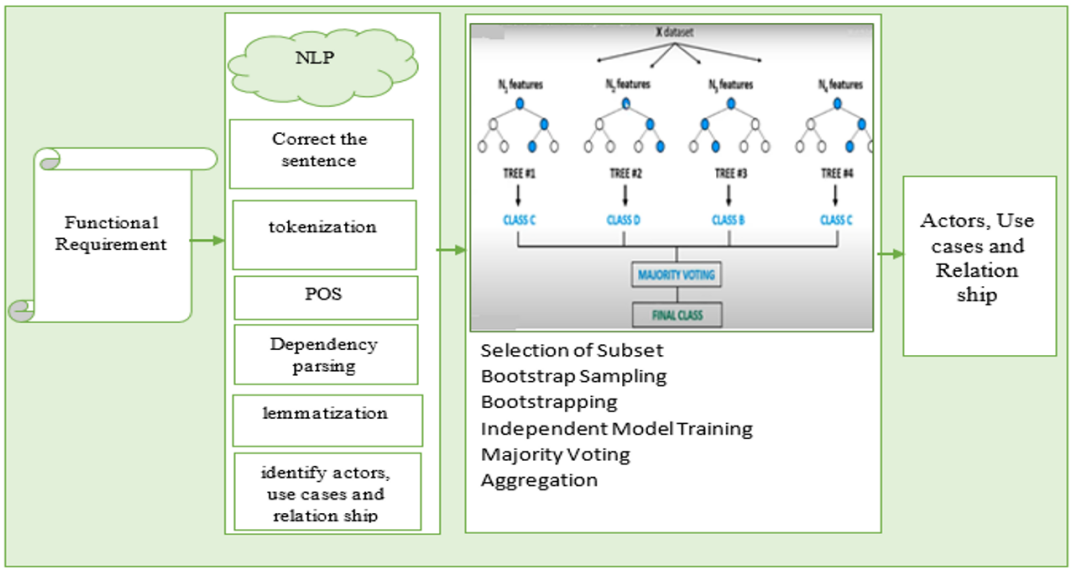

Figure 2. The developed methodology

5. CASE STUDY AND VALIDATION OF THE PROPOSED METHOD

As a case study, the suggested algorithm was utilized to satisfy functional criteria for a library management system according to the scope.

Users who register themselves in a library system are regarded as either students or staff members. A user asks for a new book tailored to their specific requests. After the request, the user checks out the book, which means that no other users can request it. Now that the customer has the option to renew a book, they might have a new due date for the book of their choice. The user is fined if they accidentally fail to give the book back by the time of the deadline. Within this structure, librarians play a crucial role. The librarian updates the library database with every student's or user's information whenever a book is checked out, taken back, or fines are paid. The librarian removes a student's record if the student leaves the university. The record for that specific book is removed if it is no longer available in the library. One of the librarian's important roles is to update the database."

When the proposed method takes the scope and enters it into the NLP, the first step will be to clean and tokenize the statements and their parts as follows: "A user who registers himself in the system is considered a student or staff member and will have applied POS, Dependency parsing, and lemmatization," as shown in Figure 3. Finding out what NLP is used for the first statement:

Actor = user
Use case = registers himself in the system
Relationship = user-> registers himself in the system

After applying NLP to all sentences, all outputs of NLP are entered as input in the RF algorithm.

The RF algorithm converts the text into numerical features. In this proposed method, the first step uses TF-IDF vectors to measure the importance of words. It then uses a semantic similarity score (Word2Vec, Bidirectional Encoder Representations from Transformers (BERT), cosine similarity) and POS tagging features to verify the subject-noun structure (subject + subject) and calculate a length/symbol count, which helps detect duplicates. Each requirement becomes a feature vector, and then training the RF Class 0 = Irrelevant / Duplicate, Class 1 = Relevant / Unique will be started. RF creates multiple decision trees on subsets of features. For each tree, votes for the majority win to classify the requirement. After applying RF on the output NLP, the classification of statements is shown in Table 3.

Based on the results, two actors and six use cases were extracted. The diagram of use cases will be drawn manually using Enterprise Architect version 16, where the final use case of the library management system is shown in Figure 4.

Table 3. Classification of statements using the Random Forest algorithm

Requirement	Classification	Action
Register himself in the system	Relevant	Keep
Request the book	Relevant	Keep
Check out the book	Duplicate	Merge with "request"
Renew book	Relevant	Keep
Receive a new due date for the book of their choice	Duplicate	Merge with "Renew"
Return the book	Relevant	Keep
Give book back	Duplicate	Merge with "Return"
Pay fine	Relevant	Keep
Update database	Relevant	Keep

tokenization	user who Register himself in system is considered a staff member or student
POS	Noun: (User, Staff) Student: (Library, System) Verb: (registers, regarded) pronoun: (who, himself) adjective: (new) conjunction: (or) prepositions: (as, for)
Dependency parsing	Nsubj (registers, who) means: "who" is the nominal subject of the verb "registers". dobj(registers, himself) means: himself" is the direct object of the verb "registers"
lemmatization	Registers → register, Is→ be, Regarded → regard
For first sentence	Actor= user, use case= registers himself in the system relationship= user-> registers himself in the system
Apply NLP on all sentence	
Apply the random forest algorithm	
The Actors of the system	User, Librarian
The Use cases of the system	registers himself in the system, request the book, renew the book, return the book, pays a fine, Updating the database
The relationship between parts of the system	user →registers himself in the system, user →search and check out book, user → renew book, user → return book, user → pat fine, Librarian → update database

Figure 3. The details of the proposed technique on the library management system

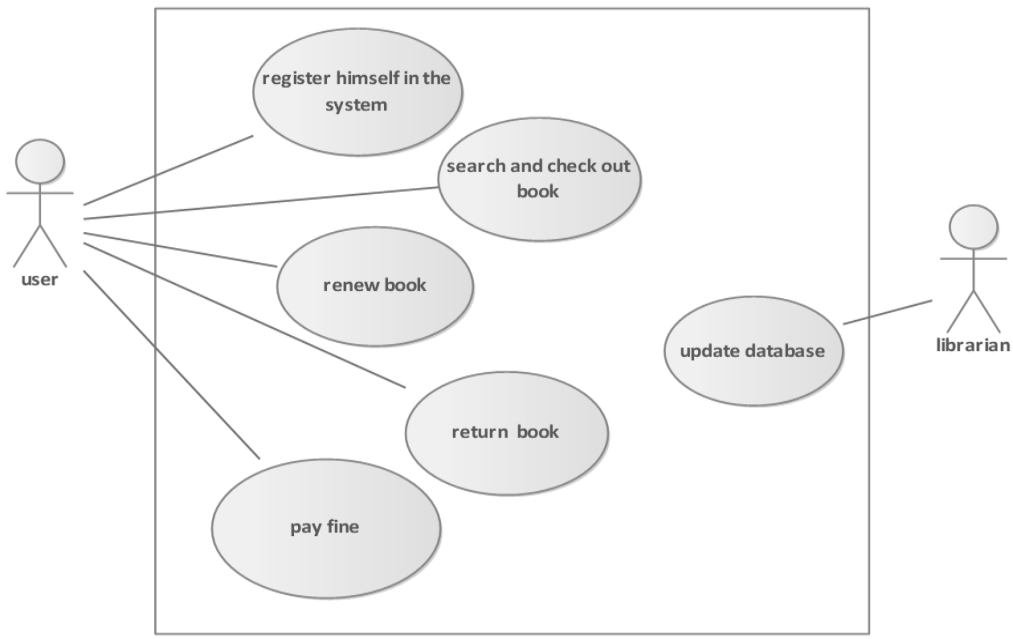

Figure 4. Use case diagram for library management system

To evaluate the current technique (NLP and RF), the proposed algorithm was applied to databases supervised by the Italian National Research Council (ISTI-CNR) [41] that contain:

1. The E-Store Project

The project contains 62 functional requirements, distributed across 20 logical groups (e.g., account management, shopping cart operations, payment processing). The requirements were

extracted from the original project documentation, focusing on identifying "actors" and "use cases." Consistency was maintained through rule-based validation, where each requirement follows a fixed format: [subject + verb + object]. This set represents a multi-class classification, providing a ratio of 3.1 requirements per class, thus testing the model's ability to differentiate between closely related classes.

2. The Inventory System

The Total Functional Requirements are 25 and categorized into 11 groupings (e.g., Stock Tracking, Low-stock Alerts, Supplier Management). With only 2.2 requirements per group, this system represents a challenging "few-shot learning" scenario. A model trained on this must generalize from extremely limited examples per class.

Annotation focused on entity-relationship extraction. Labels were applied to identify the "Object" (the inventory item) and the "Action" (update, delete, add), as the system follows a strict CRUD (Create, Read, Update, Delete) logic, making the labels highly predictable and discrete.

3. Philips System (Medical/Health Electronics)

Total Functional Requirements are 24 distributed across 19 groupings. This is the most granular system in the set. Almost every requirement represents its own unique functional group. This is used in ML to test uniqueness detection and outlier detection, as there is almost no overlap between categories. Consistency is verified against the ISO/IEC 29148 standard for requirements engineering, ensuring that the terminology (e.g., "shall," "must") is labeled uniformly across the 19 groups.

4. Total Functional Requirements are 23.

Unlike the others, these are often treated as a singular functional block or analyzed for Non-Functional Requirement (NFR) cross-cutting concerns (like privacy and security). Used primarily for binary classification (Functional vs. Non-Functional) rather than multi-group categorization. Annotations were derived from the classic Sommerville Software Engineering case study. The labels focus on "Patient Privacy" and "Clinical Safety" tags.

Because this is a well-known pedagogical system, labels have been vetted by multiple researchers in the NLP4RE community, leading to high inter-annotator agreement, as shown in Table 4.

The outcomes, using several measures including recall and precision, were contrasted with the findings of the study [9].

According to Table 5 and Figures 5 and 6, recall shows the percentage of all actual positives that have been correctly identified, whereas precision shows how well the positive predictions performed.

The introduced framework was evaluated with the approach followed in applying a machine learning framework based on NLP [18] on users' stories. The result of the accuracy metric is equal to 75.52% for research [18], while the accuracy metric of the proposed method is equal to 77.8%.

Table 4. Description of the data set

System	Functional Requirements	Groups	Average Req per Group	Primary ML Use Case
E-Store	62	20	3.1	Multi-class Classification
Inventory	25	11	2.2	Few-shot Learning
Philips	24	19	1.2	Fine-grained Entity Extraction
MHC-PMS	23	N/A	Variable	Binary Classification (NFR vs FR)

Note: MHC-PMS = Mental Health Care - Patient Management System; ML = machine learning; NFR = Non-Functional Requirement; FR = Functional Requirement.

Table 5. Experimental evaluation results

System	Precision (%)		Recall (%)	
	NLP [9]	NLP+ RF Algorithm	NLP [9]	NLP+ RF Algorithm
E-Store	90%	91%	88%	89%
Inventory system	73%	75%	70%	73%
Philips	58%	60%	71%	73%
MHC-PMS	67%	68%	50%	51%
Average	72%	73.5%	69.80%	71.5%

Note: MHC-PMS = Mental Health Care - Patient Management System; NLP = Natural Language Processing; RF = Random Forest.

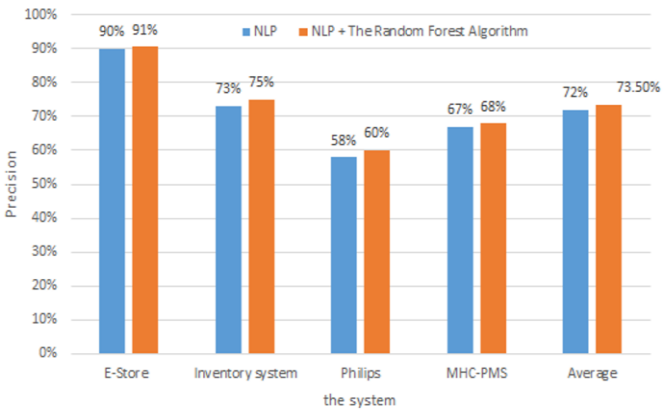

Figure 5. Chart of precision for the proposed method and the Natural Language Processing (NLP) method

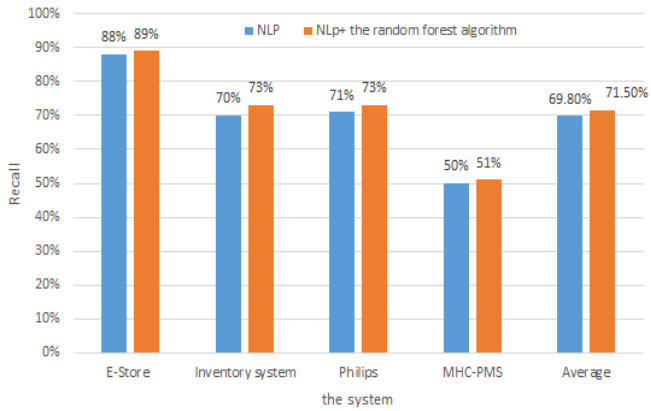

Figure 6. Chart of recall for the proposed method and the Natural Language Processing (NLP) method

Based on the analysis of the output and depending on the analysis of the difficulties that complex sentences may encounter [42], the errors encountered in the proposed study can be divided into three main categories:

1. Linguistic and grammatical errors resulting from the complexity of sentence structure, such as the problem of conjunctions and pronouns (anaphora).
2. Semantic errors: resulting from the use of synonyms or multiple names for the same entity (e.g., User vs. Staff vs. Student).
3. Filtering and merging errors: resulting from the difficulty of merging similar requirements that differ in their wording.

6. CONCLUSION

To generate a use case diagram based on predefined functional specifications, this study applies NLP techniques together with an RF classifier, which classifies the extracted data and filters out spurious outputs produced during the NLP stage of diagram construction.

By leveraging UML diagrams, this work assists engineers and software analysts throughout the requirements analysis phase. In particular, the proposed method benefits novice analysts by reducing the cost, effort, and time required for manual requirements analysis and UML element extraction. In the context of extracting actors and use cases from requirements text, precision addresses the question of whether the extracted items are indeed valid, while recall addresses the question of whether the system successfully identified most of the actual items present in the text.

When applied to several systems—including Philips, Inventory, E-Store, and the Mental Health Care - Patient Management System (MHC-PMS) system—the proposed method achieved precision values of 91%, 75%, 60%, and 73%, and recall values of 89%, 73%, 73%, and 51%, respectively, across the four systems.

The results demonstrate a significant improvement for the Inventory and Philips systems, whereas only a slight improvement was observed for the MHC-PMS and E-Store systems. This improvement can be attributed to the role of the RF algorithm in handling non-linear patterns, reducing overfitting through the integration of multiple decision trees, performing well on mixed textual and grammatical features, and refining NLP output by focusing on the most influential features.

Despite these promising results, the proposed approach has several limitations. First, it generates use case diagrams only for functional requirements expressed in English. Second, the model is capable of detecting correlations but not complex relationships, such as inclusion and exclusion associations between use cases. Third, the approach was evaluated on a relatively small dataset, which may limit the generalizability of the findings.

Future research could explore the integration of NLP with other machine learning methods, with the optimal approach to be determined through comparative analysis. It is also recommended that the proposed technique be extended to generate additional UML diagrams, such as class and activity diagrams. Furthermore, automating the process of UML diagram generation represents a promising direction for future work.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to the University of Mosul, College of Computer Sciences and Mathematics, for providing the facilities and support that contributed to the completion of this work.

REFERENCES

[1] Maatuk, A.M., Ali, M.A., Aljawarneh, S. (2015). Translating relational database schemas into object-based schemas: University case study. Recent Patents on Computer Science, 8(2): 122-131. https://doi.org/10.2174/2213275908666150710174102

[2] Dabdawb, M.M.A. (2024). Unified modeling language quantitative measures based on a behavioural model. Journal of Education and Science, 33(1): 90-98. https://doi.org/10.33899/edusj.2024.145662.1416

[3] Ying, A.T.T., Murphy, G.C., Ng, R., Chu-Carroll, M.C. (2004). Predicting source code changes by mining change history. IEEE Transactions on Software Engineering, 30(9): 574-586. https://doi.org/10.1109/TSE.2004.52

[4] Ahmed, S., Ahmed, A., Eisty, N.U. (2022). Automatic transformation of natural to unified modeling language: A systematic review. In 2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA), Las Vegas, NV, USA, pp. 112-119. https://doi.org/10.48550/arXiv.2204.00932

[5] Geetha, S., Anandha, G.S., Kumar, S. (2016). Automatic domain knowledge extraction from requirements specification text. Research Journal of Applied Sciences, Engineering and Technology, 12(9): 926-932. https://doi.org/10.19026/rjaset.12.2811

[6] Abdelnabi, E.A., Maatuk, A.M. (2021). Generating UML use case and activity diagrams using NLP techniques and heuristics rules. In the 3rd International Conference on Data Science, E-learning and Information Systems, Almaty, Kazakhstan, pp. 1-6. https://doi.org/10.1145/3492547.3492612

[7] Btosh, E., Hamad, M. (2015). Generating ER diagrams from requirement specifications based on natural language processing. International Journal of Database Theory and Application, 8(2): 61-70. https://doi.org/10.14257/ijdta.2015.8.2.07

[8] Yin, P. (2022). Learning structured neural semantic parsers. Doctoral dissertation, Carnegie Mellon University.

[9] Hamza, Z.A., Hammad, M. (2019). Generating UML use case models from software requirements using natural language processing. In 2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO), Manama, Bahrain, pp. 1-6. https://doi.org/10.1109/ICMSAO.2019.8880365

[10] Al-Msie'deen, R., Blasi, A.H., Alsuwaiket, M.A. (2021). Constructing a software requirements specification and design for electronic IT news magazine system. International Journal of Advanced and Applied Sciences, 8(11): 104-118.

[11] Fadhil, A.A., Albayati, A.H. (2025). Automatic evaluation of the student's activity diagrams in the educational process. Ingénierie des Systèmes

d'Information, 30(8): 2149-2156. https://doi.org/10.18280/isi.300820
[12] Elallaoui, M., Nafil, K., Touahni, R. (2018). Automatic transformation of user stories into UML use case diagrams using NLP techniques. Procedia Computer Science, 130: 42-49. https://doi.org/10.1016/j.procs.2018.04.010
[13] Meng, Y., Ban, A. (2024). Automated UML class diagram generation from textual requirements using NLP techniques. JOIV: International Journal on Informatics Visualization, 8(3-2): 1905-1915. https://doi.org/10.62527/joiv.8.3-2.3482
[14] Jebril, E.M., Imam, A.T., Al-Fayuomi, M. (2018). An algorithmic approach to extract actions and actors (AAEAA). In Proceedings of the International Conference on Geoinformatics and Data Analysis, Prague, Czech Republic, pp. 13-17. https://doi.org/10.1145/3220228.3220247
[15] Vasques, D.G., Santos, G.S., Gomes, F.D., Galindo, J.F., Martins, P.S. (2019). Use case extraction through knowledge acquisition. In 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, pp. 624-631. https://doi.org/10.1109/IEMCON.2019.8936279
[16] Malik, M.I., Sindhu, M.A., Abbasi, R.A. (2023). Extraction of use case diagram elements using natural language processing and network science. PLOS ONE, 18(6): 0287502. https://doi.org/10.1371/journal.pone.0287502
[17] Dias, R.P., Vidanapathirana, C.S.L., Weerasinghe, R., et al. (2023). Automated use case diagram generator using NLP and ML. arXiv preprint arXiv:2306.06962. https://doi.org/10.48550/arXiv.2306.06962
[18] Ferrari, A., Abualhaija, S., Arora, C. (2024). Model generation with LLMs: From requirements to UML sequence diagrams. In 2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW), Reykjavik, Iceland, pp. 291-300. https://doi.org/10.1109/REW61692.2024.00044
[19] Joshi, S., Summers, J.D. (2015). Requirements change: Understanding the type of changes in the requirements document of novice designers. International Journal of Mechanical Engineering Education, 43(4): 286-304. https://doi.org/10.1177/0306419015612348
[20] Majava, J., Nuottila, J., Haapasalo, H., Law, K.M.Y. (2014). Customer needs in market-driven product development: Product management and R&D standpoints. Technology and Investment, 5(1): 16-25. https://doi.org/10.4236/ti.2014.51003
[21] Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, London, England, United Kingdom, pp. 1-10. https://doi.org/10.1145/2601248.2601268
[22] Shah, T., Patel, S.V. (2016). A novel approach for specifying functional and non-functional requirements using RDS (requirement description schema). Procedia Computer Science, 79: 852-860. https://doi.org/10.1016/j.procs.2016.03.083
[23] Alashqar, A.M., Elfetouh, A.A., El-Bakry, H.M. (2015). Requirement engineering for non-functional requirements. International Journal of Information and Communication Technology, 5(2): 21-27.
[24] Shankar, P., Morkos, B., Yadav, D., Summer, J. (2020). Towards the formalization of non-functional requirements in conceptual design. Research in Engineering Design, 31(4): 449-469. https://doi.org/10.1007/s00163-020-00345-6
[25] Li, Y., Thomas, M., Liu, D. (2021). From semantics to pragmatics: Where IS can lead in NLP research. European Journal of Information Systems, 30(5): 569-590. https://doi.org/10.1080/0960085X.2020.1816145
[26] Long, M., Wang, Y., Peng, Y., Huang, W. (2022). A review of the research on the evaluation metrics for automatic grammatical error correction system. Mobile Information Systems, 2022(1): 5998948. https://doi.org/10.1155/2022/5998948
[27] Webster, J.J., Kit, C. (1992). Tokenization as the initial phase in NLP. In COLING 1992 Volume 4: The 14th International Conference on Computational Linguistics, pp. 1106-1110. https://doi.org/10.3115/992424.992434
[28] Haspelmath, M. (2001). Word classes and parts of speech. International Encyclopedia of the Social & Behavioral Sciences, 2001: 16524-16538. https://doi.org/10.1016/B0-08-043076-7/02959-4
[29] Li, H., Mao, H., Wang, J. (2022). Part-of-speech tagging with rule-based data preprocessing and transformer. Electronics, 11(1): 56. https://doi.org/10.3390/electronics11010056
[30] Singkul, S., Woraratpanya, K. (2019). Thai dependency parsing with character embedding. In 2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE), Pattaya, Thailand, pp. 1-5. https://doi.org/10.1109/ICITEED.2019.8930002
[31] Pramana, R., Subroto, J.J., Gunawan, A.A.S. (2022). Systematic literature review of stemming and lemmatization performance for sentence similarity. In 2022 IEEE 7th International Conference on Information Technology and Digital Applications (ICITDA), Yogyakarta, Indonesia, pp. 1-6. https://doi.org/10.1109/ICITDA55840.2022.9971451
[32] Kaur, H., Singh, G. (2023). A systematic review of NLP techniques: Pre-processing, feature extraction and classification. Journal of Computer Science and Technology Studies, 5(1): 34-45. https://doi.org/10.32996/jcsts.2023.5.1.5
[33] Salman, H.A., Kalakech, A., Steiti, A. (2024). Random forest algorithm overview. Babylonian Journal of Machine Learning, 2024: 69-79. https://doi.org/10.58496/BJML/2024/007
[34] Schonlau, M., Zou, R.Y. (2020). The random forest algorithm for statistical learning. The Stata Journal, 20(1): 3-29. https://doi.org/10.1177/1536867X20909688
[35] Sun, Z., Wang, G., Li, P., Wang, H., et al. (2024). An improved random forest based on the classification accuracy and correlation measurement of decision trees. Expert Systems with Applications, 237(B): 121549. https://doi.org/10.1016/j.eswa.2023.121549
[36] Pantic, I.V., Pantic, J.P., Valjarevic, S., Corridon, P.R., Topalovic, N. (2025). Artificial intelligence–based approaches based on random forest algorithm for signal analysis: Potential applications in detection of chemico-biological interactions. Chemico-Biological Interactions, 406: 111624. https://doi.org/10.1016/j.cbi.2025.111624
[37] Jaiswal, J.K., Samikannu, R. (2017). Application of random forest algorithm on feature subset selection and

classification and regression. In 2017 World Congress on Computing and Communication Technologies (WCCCT), Tiruchirappalli, India, pp. 65-68. https://doi.org/10.1109/WCCCT.2016.25

[38] Abellán Mulero, J., Mantas Ruiz, C.J., García Castellano, F.J., Moral García, S. (2018). Increasing diversity in Random Forest learning algorithm via imprecise probabilities. Expert Systems with Applications, 97: 228-243. https://doi.org/10.1016/j.eswa.2017.12.029

[39] Paucar, I.R., Yactayo-Arias, C., Andrade-Arenas, L. (2025). Random forest model based on machine learning for early detection of diabetes. International Journal of Advanced Computer Science and Applications, 16(6): 1051. https://doi.org/10.14569/IJACSA.2025.01606103

[40] National Research Council of Italy. (2019). Fmt.isti.cnr.it. http://fmt.isti.cnr.it/nlreqdataset/.

[41] Saini, A., Mussante, G., Jensen, K.L. (2024). Exploring the capability of large language models in software modeling: The case of UML diagrams. In Proceedings of the 2024 46th International Conference on Software Engineering: Software Engineering in the Age of AI (ICSE-AI), pp. 112-123. https://doi.org/10.1145/3639474.3640061