Automating Requirements Engineering using Machine Learning

Tahmina Akter
Department of Computer Science and Engineering
Brac University, Dhaka
Bangladesh

Md. Mehedy Hasan Abid
Department of Computer Science and Engineering
Brac University, Dhaka
Bangladesh

Rubaya Neshat Tanna
Department of Computer Science and Engineering
Brac University, Dhaka
Bangladesh

Jahidul Hasan Masud
Department of Computer Science and Engineering
Brac University, Dhaka Bangladesh

Mahbubur Rahman Noyon
Department of Computer Science and Engineering
Brac University, Dhaka Bangladesh

ABSTRACT
Machine learning (ML) algorithms have proven effective in automating processes across various domains, including software engineering. One of the earliest and most critical phases of software development is requirements engineering (RE), which often involves manual tasks prone to human error and inefficiency. This research aims to enhance and automate the requirements engineering process by integrating advanced machine learning techniques, particularly in the areas of natural language processing (NLP) and pre-trained models such as BERT. By leveraging these technologies, we seek to reduce development costs, improve accuracy, and streamline the transition from requirements gathering to system design. The study involves the application and evaluation of a range of machine learning algorithms to determine the most suitable approaches for automating RE tasks. Finally, we propose a set of evaluation metrics to assess the effectiveness and practicality of the developed methods in real-world software requirement specification scenarios.

General Terms
Algorithms, Design, Experimentation, Measurement, Documentation, Performance.

Keywords
Machine Learning, Requirements Engineering, Software Engineering, Natural Language Processing (NLP), BERT, Automation, Requirement Classification, Requirement Specification.

1. INTRODUCTION
Requirement engineering offers language and technology to bridge the gap between informal, imprecise, and ambiguous user requirements. In successive software development phases, these formal requirements build software systems to meet potential users' needs[1]. Requirements Engineering Process consists of Requirements elicitation, Requirement specification, Requirements verification and validation, Requirements management. By using this process, a system analyst generates system specifications. System specification helps to achieve user requirements. Machine learning helps to produce output by learning through previous data. The use of machine learning in requirement engineering can reduce the error. Since software requirement specification is primarily in natural language, so natural language processing algorithms can play a great role in processing the software requirements specification.Neural networks work like the human brain and nerve system. This technology finds patterns in the data by itself. So, a neural network can be used to find patterns in the SRS data.

For not defining practical system requirements, over 25 percent of all software projects still fail entirely and also cost billions of dollars to organizations. Requirement Engineering is a crucially important aspect of software engineering. Errors produced at this stage, if undetected until a later stage, can be very costly [7]. Machine learning methods should be used for requirement engineering. But there is not a lot of work automation in this section. Most of the research on requirement engineering is on classification between functional and nonfunctional requirements. If automation can be used, then it will make the software development process faster. There are already existing methods that perform the requirement gathering process, and the system analyst employs them to gather requirements, but they continue to encounter numerous issues. This research attempts to use machine-learning algorithms to build a framework that will generate software requirement specifications based on user requirements. To train the algorithm, existing SRS data will be collected from different sources. After that, those requirements will be processed using NLP libraries and will be used to train a neural network.

In this research paper, a comprehensive analysis of related works is first presented. The literature review and background of NLP, machine learning algorithms, and the BERT model applied are discussed in the following chapter. The section on Dataset Analysis covers data collection, data cleaning, and data organization, providing a detailed description of the software requirements specification (SRS) data gathered from various sources. The subsequent processing of these requirements using NLP libraries is then explained, leading to the generation of requirements for a user-defined system. The Implementation chapter describes the procedure followed for model development, while the Conclusion section summarizes the research findings and provides recommendations for future work.

2. LITERATURE REVIEW
Several previous research studies relevant to this work have been reviewed. Below are some key examples of prior projects that have contributed to the understanding of the field:

The Authors Winkler, J., Vogelsang, A.[10] introduced a Natural language requirements specification that is a type of requirement specification that is frequently used to capture the results of the requirements engineering process. Additional information, such as explanations, summaries, and statistics, can be found in these papers. This paper describes a method for classifying content items in a natural language requirements specification as requirements or information automatically. They employed a set of 10,000 content pieces taken from 89 criteria specifications of our industry partner to train the neural network. Our method achieves a steady classification accuracy of 81 percent by using 90 percent of the content pieces as training data and 10 percent as test data. The Authors used a single link text clustering technique on the dataset to increase its quality. Within big clusters, the author manually changed the classification of incorrectly categorized items. The resulting dataset was unbalanced, with requirement content components nearly five times greater than information content elements. After training the neural network, the method can accurately categorize new requirements documents, comparable to the accuracy of CNNs used for other tasks.

Zhao et al.[19] introduced NLP as a theoretically grounded set of computer approaches for evaluating and modeling naturally occurring texts at one or more levels of linguistic analysis in order to achieve human-like language processing for a variety of tasks or applications. The majority of those who took part in the poll said that NL was commonly used at their workplaces. Used to describe and specify software and system requirements They extracted data for the publication facet's categories and subcategories in Phase 1. The data for each of the remaining three aspects was removed in Phase 2. They carried out thematic synthesis on the descriptions of input document kinds in Phase 3. They also did thematic synthesis on descriptions of NLP techniques, NLP tools, and NLP in general. To make certain that our study selection was as accurate as possible, free of researcher bias and human mistake. They used a strict study selection approach that was led by well- specified inclusion and exclusion criteria and enforced by crosschecking and independent checking of selected and deselected studies. The first systematic mapping investigation of the landscape of NLP4RE research was published in this article. The mapping study includes 404 primary studies from 11,540 search results, which were rigorously analyzed. This mapping study demonstrates how far NLP4RE research has come in the last 15 years, especially in terms of publishing and tool development. There is now a palpable sense of anticipation that NLP4RE research will soon be translated into a useful tool to aid RE practice.

Because goals that are more realistic are more likely to lead to future disappointment, Ryan, K.et al. claim in their research study [2] that the potential importance of natural language processing in the Requirement engineering process has been overstated in the past. The system is thought to be viable and desired, and it would make requirements engineering specification easier and more precise. A system that generates sample case scripts for the client's approval. The topics could be selected to represent both extreme (limited) and ordinary (anticipated) situations. One source of these misunderstandings about NLP could be the perception of RE as primarily a difficulty in interlanguage communication. For two reasons, this isn't feasible. For starters, there are many languages to study rather than just one. Second, and most importantly, other professionals' clients (e.g., lawyers, architects) rely on them to comprehend their wishes and convert them into specialist jargon. They avoid claims of computers that will "understand" language in any meaningful way for all of these reasons. The complexity of large-scale systems is a reflection of their inherently complex nature, rather than a result of properly and completely defining them. We can expect plans to be mathematically described and confirmed in terms of technical performance. Nonetheless, their adherence to need will be assessed throughout time in a fluid and mostly undefined social setting.

Dalpiaz et al. proposed in another study [13] that even stakeholders with limited knowledge in requirements engineering may write and comprehend NL requirements. Furthermore, manually examining large collections of NL requirements to get an overview, detect inconsistencies, redundancies, and missing prerequisites is difficult. The goal is to conduct significant research on the application of NLCP tools and techniques in RE practice, as well as to evaluate requirements-related documents automatically. NLP is rapidly becoming a foundational technology in a variety of fields and applications. The challenge now is one of sustainability. They intend to hold NLP4RE in the future. In the years to come, they will be looking for a stronger integration with other communities. They discussed holding a workshop as an event of a conference such as the Association of Computational Linguistics (ACL), the International Conference on Computational Linguistics (COLING), or the Empirical Methods in Natural Language Processing (EMNLP) in 2019. (EMNLP).

Dias Canedo et al. investigated textual function extraction techniques and machine learning algorithms to respond two significant queries: "Which fits best for categorizing Software Requirements into Functional Requirements and Non-Functional Requirements, and the subclasses of non-functional Requirements. In the paper [16], the research was conducted using the PROMISE exp dataset, a freshly constructed dataset that enhances the already known PROMISE repository, a repository providing software requirements. Logistic Regression, Support Vector Machine, Multinuclear Naive Bayes, and k-Nearest Neighbors were the classification techniques employed. They looked how to improve the classification of system requirements and evaluate which text vectorization techniques, such as Word Bag, Term Frequency and Inverse Document Frequency, and Chi-Squared, are the most efficient, and which learning algorithm has the best performance in the task of classifying requirements. They use the PROMISE exp database to evaluate the combination of various techniques, increasing the PROMISE database. They discovered that TF-IDF and LR together had the best performance metrics for binary classification, non-functional classifications, and requirements in general, with an F-size of 91 percent for binary categories, 74 percent for 11-granularity classification, and 78 percent for 12. The findings of the research can be used as a reference or guideline for future study by developers who want to automate a wide range of software needs. Researchers in this field as a guideline for future research can also use it.

The author Nazir et al. [11] discovered a method for extracting the elements of interest from raw plain text documents automatically. As a result, it is used to refine and eliminate acceptable system requirements from natural language artifacts. The software requirements are gathered and written in plain text in a human-readable natural language. Such linguistic criteria, on the other hand, are of little use to technical stakeholders. As a result, it is critical to fine-tune the initial requirements in order to get the most out of them. They devised a review methodology that includes six categories for choosing 27 research, as well as selection and rejection criteria. For the

search, they utilized precise phrases connected to the subject. To narrow the search results, they used several criteria. They discovered that NLP approaches produce positive results when it comes to extracting relevant aspects from plain text software requirements. At lower levels of NLP, such as tokenization and POS tagging, however, a few human steps are frequently required. As a result, it is difficult to predict that NLP totally automates requirement refinement from raw text. However, the suggestion of the most up-to-date instruments in this regard is advantageous.

Machine learning methods have been demonstrated to have substantial functional importance in various application domains, according to Iqbal et al. in their work [14]. This is especially true in domains where large databases are available. The engineering of requirements is an important part of software engineering. ML can be advantageous by simulating human processing. DOORS, a free-form text-based tool with lightweight structural features, has become the standard in practice. They have shown that ML has the potential to be a cornerstone in RE. For the time being, it appears that the domain is undergoing a pre-scientific process. They ask for a more comprehensive survey to confirm the tentative conclusions presented in this paper. The stakes are really high. While requirements engineering is currently a topic of intense research, academic attempts to address its issues have yielded few practical outcomes.

Parra et al. proposed in their research paper [9] Low-quality requirements might lead to mistakes throughout project development. If low-quality needs are not recognized in a timely manner, they are regarded as the most expensive to rectify. The following are the primary aspects that give tools for requirement management: Validation, storage management, and traceability Quality Management The method tries to construct classifiers using induction rule-based machine learning techniques. By altering the learning cases or discovering various ways of applying them, the classifiers' accuracy can be increased. The proposed changes are to analyze the same requirements using classifiers.

The software engineering presented by Ning et al. [4] is now one of the key research points in the area of software engineering. Prior to beginning development, the author's focus of issue analysis is to obtain a better knowledge about the situation. In the article, the primary role is to bridge the communication gap between the user and the system admin. RE (Requirement Engineering) of objectification and modeling facilitated by MOR Editor. It may be utilized as a guide for implementing actual RE procedures. This is solely appropriate for functional requirement needs. To utilize this paradigm, users must have fundamental knowledge of software engineering.

Hayes et al.[8] argued that in RE there is a set of issues that lend themselves nicely to ML approaches. In the paper, it shows Weka is a set of classification trees, which are directed instructional methods. The author proposes two uses of the component as a first idea validation. Trace Labs WekaClassifiersTrees TraceLab makes categorization easy, efficient, and repeatable. The author also thinks that a number of additional RE issues may benefit from this aspect. They intend to provide more elements to make it easier to utilize TraceLab for a wide range of RE issues.

Oster et al. argued in article [6] that the main objective of RE is an effective method for defining requirements for a system. For the method to be satisfactory, the functionality defined by the model's purposes should be implemented. Preferences over soft goals are important in the main objective of RE. The Preferred Reasoned finds the goal assignments that are most favored for an objective structure. From a text input file, the Objective Concept Analyzer builds the target framework. So, the framework has the potential to greatly enhance the documentation and use of design preferences.

From another research article, we get to know that the authors Zhu et al. [3] developed a RAAS framework to assist the RE process, a report that researched automated implementation details for RE. RASS supports the decomposition of the formal specification through knowledge acquisition tools. The RASS project aims to decompose the problem of specifying a large-scale complicated software into several much simpler and smaller-scale problems. Two key problems must be solved to achieve the practical usability of the proposed approach. This section discusses the RASS' solution to these problems. Experimenting with an automated framework for RE at the requirement stage can result in substantial automation.

The RE process, according to the authors Jiang et al. [5], is an integral element of the whole software lifecycle and plays a significant role in maintaining the performance of the overall system. The advantages of RE are now well documented in the journals. Several approaches are used to address various parts of the RE process and system. In this paper, there were three case studies done. In this whole development; the author has been highly involved. The analysis and categorization of RE methods have made great progress thanks to this research paper.

Alessio Ferrari et al. [15] proposed a natural language processing strategy for identifying ambiguous terms across domains and ranking them by ambiguity score in another work. The strategy relies on the creation of domain-specific language models for each stakeholder. They tested the method on seven different elicitation scenarios involving five different fields. Ambiguity is mostly studied in written NL requirements; nevertheless, because the focus in spoken NL is on requirements elicitation meetings per form, it is useful to refer to recognized classifications of ambiguity in written requirements. The skip-gram with negative sampling (SGNS) method, which is implemented in the word2vec software package, is used to create word embedding based on Harris' distributional hypothesis. The task's first purpose is to compare the similarity of the automatically generated rank (sample ranking) to the humanly generated one (ground-truth ranking). This assignment tries to explain ranking mistakes of two types: (a) items that are rated lower in the ground-truth order and appear in H in the sample ranking; and (b) elements that are ranked higher in the ground-truth order but appear in H in the sample ranking. (b) In the sample ranking, features that are higher in the ground-truth ranking appear in L. They compare the ambiguity rankings generated automatically with those obtained manually by the authors and many annotators hired through Amazon Mechanical Turk in the evaluation. The approach generates a scale with a maximum Kendall's Tau of 88 percent. Two individuals to ensure the authenticity of the annotations made on the sentences during the Manual Annotation task carry out the annotation process independently. Cohen's Kappa is used to calculate inter-rater agreement (Lan- dis and Koch 1977). The Ground-Truth Ranking is based on (a) the averages of the scores supplied by different annotators and (b) the averages of the scores received by three separate sentence sets comprising the same phrase. However, the application of the technique proved ineffective in terms of performance for numerous elicitation circumstances. Their ultimate goal is to use the current research's findings in

real-world elicitation scenarios. This assessment is planned as a long-term goal due to the design's complexity, and it should come after our short-term future effort aimed at fine-tuning the method.

Peinelt, N., Nguyen, D., and Liakata, M. [18] present a new topic-informed BERTbased architecture for pairwise semantic similarity detection in their work. Across a number of English language datasets, the BERT model performs better in terms of acceptable and proper baselines. Until recently, language models could only read text input in one of two ways: left-to-right or right-to-left. BERT is one of a kind in that it can read in both directions at once. Bi-directionality is the term for this capability, which was made available by the development of Transformers. The tight integration of NLP and SbSE is a crucial strategy, with data gained over time being used to improve the system with each consecutive iteration of the requirements specification.

According to Lash, A. et al research the majority of requirements are written in NL. Many concerns, such as ambiguity, specification issues, and incompleteness, must be reported. These problems are divided into three categories: word, word sentence, and document. It uses tools to compare requirements statements according to their grammatical subject. The program (FMTV) will analyze an example set of criteria from a family of military tactical vehicles. The device aims to show how requirement analysis may address the semantic processing level. It can be used to identify specific areas in need of more research and development. Many difficulties, such as ambiguity, specification challenges, and completeness, are inherent with writing requirements in NL. An objective and reproducible way of analyzing requirements is provided by a linguistic approach. NLP approaches, on the other hand, can be used to automate the analysis process [12].

S. Panichella et al. [17] proposed The ability to collect feedback from end-users and the success of requirements engineering (RE) sessions are both associated with software quality. Requirements-Collector, a tool for automating requirements specification and user input analysis, was suggested in this paper. Machine learning (ML) and deep learning (DL) computational processes are used in the tool. According to the research, it can reliably classify RE requirements and user review feedback. The paper argues that it has the potential to transform the work of software analysts, resulting in a significant reduction in manual activities, improved collaboration, and a greater focus on analytical tasks. They also introduce Requirements-collector, a tool that automatically classifies requirements using machine learning and deep learning. Preliminary results show that it is accurate at extracting requirements. The findings can be used to determine how appropriate ML and DL models are for achieving high accuracy.

3. PROPOSED SYSTEM MODEL

3.1 PROBLEM STATEMENT

To build the desired framework learning model that will generate the requirements for the system, at first, some initial requirements from the user need to be taken. For example, the user will give the end-to-end feature that the user wants from the system. The user will then provide the project domain, like what kind of system the user wants to build. For example, E-commerce system, Online Banking System, etc. After that, the user would be asked for the scalability of the system. To meet the desired software requirements, a budget is also essential. So, the user will then be asked for the budget. Then, the system will generate the system features, including both functional and non-functional requirements, from the user input.

Formal model of the learning algorithm will be:

1. Domain set: An arbitrary set, X which contains system domain, D, scalability S, budget B set: The label set, Y will contain three-element. Those are system features (Sf1 ,Sf2 ,....Sfn), functional requirements (Fr1 ,Fr2 ,....,Frn), Nonfunctional requirements (Nfr1 ,Nfr2,....,Nfrn)

3.2 PROPOSED ARCHITECTURE

Fig 3.1 depicts the workflow of how the learning algorithm is trained and generates output from the existing data. To build the desired framework and learning algorithm, a large amount of software requirements specification data was needed. Using this data, the algorithm was trained, and the more data used for training, the more accurate the output became. Therefore, the data collection process was one of the main challenges in making the learning algorithm more robust. Since data was collected from different sources, the formats varied from one another. Consequently, the data needed to be cleaned so that it could fit into the learning algorithm. After that, the data was organized into different groups for training purposes.

After working with the data, it was divided into two parts. A total of 80% of the random data was used for training the learning algorithm, and the remaining 20% was used for testing the learning algorithm.

In the learning phase, sentiment analysis, lemmatization, and classification were required, for which natural language processing libraries were used. As the work mainly involved text data, it was convenient to use NLP libraries designed for text processing. For this reason, the system employed the BERT algorithm to vectorize data, as machine learning algorithms do not directly process text data. After encoding text data into vectorized form using BERT, the data was applied to cosine similarity to calculate similarity scores. Finally, based on similarity scores of features from the existing dataset, requirements were generated for the user-preferred system.

After the machine learning model was created and similarity scores were calculated, the user was asked for preferred system information. This information was then provided to the machine learning model, which generated the recommended requirements for that system.

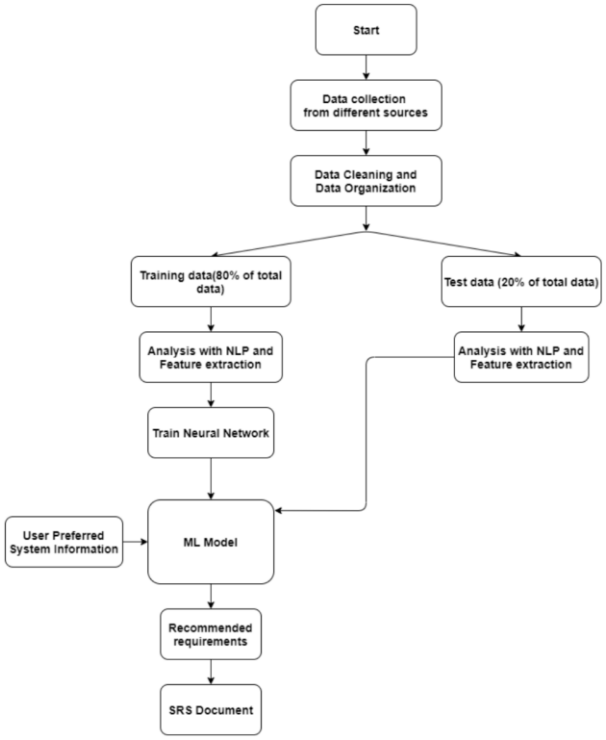

Fig 3.1: System Architecture Diagram

4. DATASET ANALYSIS

This section outlines the data collection process, as well as the steps taken for data cleaning and organization.

4.1 DATA COLLECTION

The project needed a large set of data to train the neural network so that the machine-learning model performs more accurately. For collecting software requirements specification data, several software companies were approached to obtain SRS (software requirements specification) data. Data was collected through the responses to the questions that form the basis of understanding the problem or exploring the objective's idea. Data was also collected from online open sources. The Google search engine was used to search for available datasets on the internet. Some websites were visited to collect software requirement data. Faculty members of the institution also assisted in collecting data. Some problems were encountered in collecting data, such as a few companies not wanting to share their data due to confidentiality. In addition, there are limited resources available on the internet regarding the SRS dataset.

4.2 DATA CLEANING

The dataset was collected from multiple sources. Some data were obtained from the internet, some were received via mail, and some were handwritten. Therefore, all datasets needed to be merged into a single format. After merging, the dataset contained some unnecessary and incorrect data. Consequently, incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data were removed from the dataset. The dataset then contained only properly formatted data suitable for training input. If inaccurate data were not removed, the outcomes and algorithms would be unreliable, even if they appeared correct.

4.3 DATA ORGANIZATION

After cleaning the dataset, the database was organized to allow the data to be used more effectively. The dataset was prepared to serve as input for the machine learning algorithms, and its organization was carried out accordingly. To create the training dataset, two datasets were constructed and labeled as the Project dataset (Figure 4.1) and the Feature dataset (Figure 4.2). The Project dataset contains all the project names from the SRS documents, along with a unique project ID for each project.

Project ID	Project Name
1	Online Auction System
2	E-learning website
3	Diagnosis Automation from Medical Image using Machine Learning
4	Attendance Application
5	Online Voting System
6	Online Auction System
7	House Rental Management System
8	Tourism and Travel
9	PC Builder App
10	Car Rental System
11	Movie Rating and review System
12	Blog App

Fig 4.1: Project Dataset

The Feature dataset (Figure 4.2) contains the project ID from which the corresponding requirements are derived. This project ID matches the project ID in the Project dataset. For each individual requirement, the Feature dataset includes a unique Feature ID, the name of the feature, and a description of the feature. The Feature Name represents a use case of the system, while the Feature Description elaborates on the functionality of the requirement.

Project ID	Feature ID	Feature Name	Feature Description
1	1	Registration and Login	All the users need to get registered and then login in order to bid.
1	2	Dashboard	The bidders will be able to view upcoming auction events on their dashboard or they may even search for the products.
1	3	Look and Feel	The customers will be able to view an image of the products along with their features.
1	4	Bidding Rules	The customer will bid for the products once the auction starts till it ends.
1	5	Notification	The highest bidder will get a notification email after the end of the auction and further instructions regarding payment.
1	6	Payment	The payment process will be fully verified to detect and control any fraudulent behavior.
1	7	Performance	The inventory database must be updated in real-time.
1	8	Cross-Platform Support	The system will operate in Mozilla Firefox, Google Chrome, Opera, and Safari etc.
1	9	Security	Log in attempts are limited to 3 times.
1	10	Session	The system will ensure inactive logout after a certain period.
2	11	Instructor Rules	The teachers will be able to upload videos, pdf or any other contents they want.
2	12	Instructor Rules	Once you upload a course, you can't be a student of your own course.
2	13	Passing Criteria	To pass a course a student needs a certain grade which is set by the instructor.
2	14	Functionality	The courses will also include online exams.
2	15	Look and Feel	The system will be able to let knowledge seekers search for courses by their preference.
2	16	Intelligent Search	The system will let seekers further filter the search option by course type.
2	17	Intelligent Search	Intelligent system to show course suggestions.
2	18	Availability	Our system should be available 24/7.
3	19	Admin	Admins will have the authorization of adding doctors and lab technicians information.
3	20	Usability	Doctors uses the dashboard for patient information.
3	21	Doctor	Doctors can change the diagnosis report if necessary.
3	22	Doctor	Doctors can also add the prescription in that dashboard.
3	23	Lab Technicians	Lab technicians can create information about patients and organize the report accordingly.
3	24	Doctor	Doctors can download the report from the system whenever he wants.
3	25	Data Integrity	Only the admin can access all the information of the system.
3	26	Admin	Admin can change, remove or update the initial password to doctors and technicians for security purposes.
3	27	Cross Platform Support	The system will operate on windows.
3	28	Performance	The database will be updated in real time.
3	29	Image	The system can import images.
4	30	Identify hotspot	Able to identify the student's smartphone using the hotspot of the teacher's phone.
4	31	Duration of presence	Note the duration of presence of each student.
4	32	Attendance	Mark the students based on their attendance.
4	33	Calculate attendance	Calculate the total attendance and generate a percentage value.
4	34	Teacher	The teacher will be able to set the criteria of marking.
4	35	Session	The system will tag 10 students at one time i.e. allowing 10 students to connect to hotspot at a time.
4	36	Notification	The student will be notified if the system fails to tag the students.

Fig 4.2: Feature Dataset

4.4 ENTITY RELATIONSHIP DIAGRAM

Basically, there will be three entities in the system (Figure 4.3). The first one will be a project, where the primary key will be the project ID. Then, there is another entity called Feature. In the Feature entity, Feature ID will be the primary key and project ID will be the foreign key, taken from the project ID. Lastly, there is an entity called Similarity Check. In Similarity Check, Similarity Score ID will be the primary key, while Project ID1 and Feature ID1 will be foreign keys. This will be generated through the implementing algorithm from project and feature. One project can have one or many features.

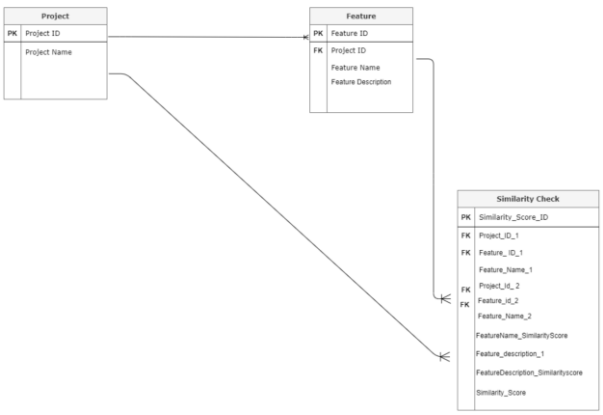

Fig 4.3: Entity Relationship Diagram

5. MODEL SPECIFICATION

In this section, the underlying models and methodologies used in the research are described in detail. Each model has been selected to address specific challenges in data processing,

representation, and similarity measurement. The specifications outlined here provide the foundation for implementing the proposed framework and ensuring accurate performance during experimentation.

5.1 Natural Language processing (NLP)

NLP is an artificial intelligence (AI) branch that is mainly focused on communication between computers and human languages. NLP refers to a computer program's capacity to interpret human language as it is spoken and written – also known as natural language. It isn't easy to teach computers the linguistics of human language, but we have made several breakthroughs in this sector in recent years. Natural language processing enables computers to interact with people in their native language and handle other language-related tasks. For example, NLP enables computers to read text, hear a voice, analyze it, gauge sentiment, and identify which aspects are essential. NLP has been around for approximately 50 years and has its roots in linguistics.

5.2 BERT

BERT is an initial for Bidirectional Encoder Representations from Transformers. BERT is an open-source NLP machine-learning framework. BERT is intended to assist computers in understanding the meaning of unclear words in the text by establishing context through the use of surrounding material. The BERT framework was trained using Wikipedia text and may be improved with question-and-answer datasets.

Historically, language models could only interpret text input sequentially – either right-to-left or left-to-right – and not both. BERT is unique in that it can read in both directions at the same time. This capacity, made possible with the introduction of Transformers, is referred to as bi-directionality.

There are three key reasons why BERT is considered highly effective. First, it is pre-trained on a large amount of data. Second, it takes into account the context of a word within a sentence. Third, it is available as open-source software, which makes it widely accessible for research and application.

In this research, data such as feature names and feature descriptions were collected from multiple research papers. Using BERT, the similarity of one document to another was evaluated. The similarity score indicates whether the meanings of two texts are similar or different. To calculate these similarities, both semantic similarity and cosine similarity (via scikit-learn) were applied. Semantic similarity measures how close the meanings of two words or texts are, even when the words are not exact matches, by leveraging Natural Language Processing (NLP) techniques such as word embedding. Cosine similarity, on the other hand, measures the similarity between two documents irrespective of their size by calculating the cosine of the angle between their vector representations.

5.3 COSINE SIMILARITY

Cosine similarity is a statistic that determines how similar texts are independent of their size. It computes the cosine of the angle formed by two vectors projected in a multi-dimensional space. The similarity measure is helpful because, even if the Euclidean distance (due to size) separates two similar documents, they might yet be closer together. The greater the cosine similarity, the smaller the angle.

$$similarity(A, B) = (A \cdot B) / (||A|| \times ||B||) = (\Sigma Ai \cdot Bi) / (\sqrt{(\Sigma Ai^2)} \times \sqrt{(\Sigma Bi^2)})$$

Fig 5.1: Formula of Cosine Similarity

6. REQUIREMENT GENERATION

This section describes the process of generating requirements using machine learning techniques. The approach relies on semantic similarity to derive new requirements from existing datasets.

6.1 ALGORITHM

```
For each project p in feature dataset:
    For each feature name f and feature description d in p
        Calculate_similarity_score(p,d,f)
Calculate_similarity_score(project,featureName,featureDescription):
    project_1=project
    project_2=any one of the projects of the project dataset excluding project_1
    project_1_Id=project_1_id from the feature dataset
    project_2_Id=project_2_id from the feature dataset
    feature_1_id=feature_1_ id from feature dataset
    input_for_featureSImilarityScore=[]
    input_for_featureSImilarityScore (feature)
    input_for_featureDescriptionSimilarityScore=[]
    input_for_featureDescriptionSImilarityScore.append(featureDescroption)
    feature_2_id_list=[]
    for each feature name f and feature description d in project_2:
        feature_2_id_list.append(f id from dataset)
        input_for_featureSImilarityScore.append(f)
        input_for_featureDescriptionSImilarityScore.append(d)

    featureNameSimilarityscore=bert_model(input_for_featureSImilarityScore)
    FeatureDescriptionSimilarityScore=
bert_model(input_for_featureDescriptionSimilarityScore)
    For each feature_2_id in feature_2_id list:
        Insert_into_similarity_score dataset(Project1Id, Project2Id,
        FeatureName1ofProject1, FeatureName1ofProject2,
        FeatureNameSimilarityScore, FeatureDescpription1ofProject1,
        FeatureDescription1ofProject2,
        FeatureDescriptionSimilarityScore,SimilarityScore=((03*
        FeatureNameSimilarityScore)+(0.7* FeatureDescriptionSimilarityScore)))
```

Fig 6.1: Algorithm

This research used semantic similarity for generating the requirements from the existing dataset. BERT model and Cosine similarity process is used to measure the semantic similarity between two sentences. To get the semantic similarity score,

Each individual project feature name was compared with the feature names of other projects. Initially, the feature names were passed to the BERT semantic similarity model, which encoded and vectorized the text data to prepare it for application in different machine learning algorithms. The vectorized data was then passed to the cosine similarity function to generate a similarity score. As a result, each feature name of a project received a similarity score when compared with the feature names of other projects.

Following the same procedure, similarity scores were also calculated for feature descriptions by comparing the descriptions of individual project features across different projects. After obtaining both the feature name similarity score and the feature description similarity score, a final similarity score was computed to represent the overall similarity between two requirements from different projects.

```
user_system_name=input ("Enter system name to get requirements")
projects=[] // a list of projects name
projects.append(user_system_name)
for each project p in project dataset:
        projects.append(p)
project_name_similarityScore=bert_modl(projects)
matched_project_index=[]
for each index in project_name_similaryScore:
        if(score[index]>=0.80):
                matched_project_index.append(index)
feature_name=[]   // A list of recommended feature name
feature_description=[]    // recommended feature description list
for each element of matched_project_index :
        project_1=project
        project_2=any one of the projects of the project dataset excluding project_1
        project_1_Id=project_1_id from the similarity_score dataset
        project_2_Id=project_2_id from the similarity_score dataset
        for each row in similarity_score dataset:
                if project_1_Id==row[Project_Id_1] and project_2_Id==row[Project_Id_2]:
                        score=row[Similarity_Score]
                        if(score>0.60):
                                feature_name.append(row[Feature_Name_1])
                                feature_description.append(row[Feature_description_1])
                                feature_name.append(row[Feature_Name_2])
                                feature_description.append(row[Feature_description_3])
for each index of feature_name list:
print("Feature Name : ",feature_name[index],"Feature Description",feature_description[index]
)
```

Fig 6.2: Algorithm (Continued)

Once the similarity check dataset calculation was completed, the system was designed to accept a system name as input from the user. The provided system name was then compared with each project name in the project dataset using semantic similarity. If the similarity score between the user-provided system name and a project name exceeded the defined threshold, that project was selected for further calculations. After identifying all projects that met the threshold condition with the user-provided system name, the corresponding requirements were generated based on the similarity scores of each project's features and their similarity with the features of other projects that satisfied the threshold condition.

Similarity_Score_Id	Project_Id_1	Feature_Id_1	Feature_Name_1	Project_Id_2	Feature_Id_2	Feature_Name_2	FeatureName_SimilarityScore	Feature_description_1	Feature_description_2	FeatureDescription_similarityScore	Similarity_Score
1	1	1	Registration and Login	2	11	Instructor Roles	0.610177577	All the users need to get registered and then login in order to bid.	The teachers will be able to upload videos, pdf or any files for students they want.	0.555454585	0.577331781
2	1	1	Registration and Login	2	12	Instructor Roles	0.610177577	All the users need to get registered and then login in order to bid.	Once you upload a course. You can't be a student or instructor of	0.413554370	0.492201663
3	1	1	Registration and Login	2	13	Passing Criteria	0.565343976	All the users need to get registered and then login in order to bid.	To pass a course a student needs a certain grade which is set by the instructor.	0.447442442	0.494603056
4	1	1	Registration and Login	2	14	Functionality	0.588112395	All the users need to get registered and then login in order to bid.	The courses will also include online exams.	0.466497272	0.533143281
5	1	1	Registration and Login	2	15	Look and Feel	0.596410692	All the users need to get registered and then login in order to bid.	The system will be able to let knowledge seekers search for courses by their preference.	0.530103750	0.596750727
6	1	1	Registration and Login	2	16	Intelligent Search	0.713651538	All the users need to get registered and then login in order to bid.	The system will let seekers furthermore filter the search option by course type.	0.550149918	0.615550566
7	1	1	Registration and Login	2	17	Intelligent Search	0.713651538	All the users need to get registered and then login in order to bid.	It will be the system to show course suggestions.	0.385803461	0.516842692
8	1	1	Registration and Login	2	18	Availability	0.588888280	All the users need to get registered and then login in order to bid.	Our system should be available 24/7.	0.456390729	0.509338152

Fig 6.3: Similarity Check Dataset

6.2 EXPERIMENTAL RESULT AND ANALYSIS

After taking the input from the user (Figure 6.4), the system will provide the suggested requirements.

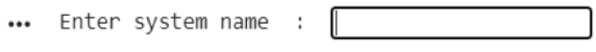

Fig 6.4: Taking input from the user

In Figure 6.5, the output shows that the system generates both functional and non-functional requirements for the user-given system based on the requirements of existing projects. Each feature contains a description according to that feature, so that the user can understand the technical aspects of the requirements. Also, as this system generates output based on existing systems, the user can get an idea of the features of other systems related to the user-given system and use those requirements.

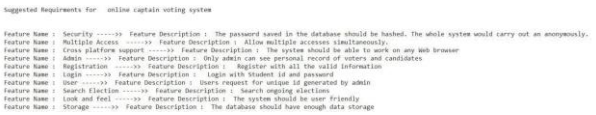

Fig 6.5: Requirement Generation

7. FUTURE PERSPECTIVES

The algorithm is still in its development stage. So far, the system checks the similarity with the project name, but in the future, it will also match similarity with the project description and consider the project volume. By observing the project volume, the system will generate requirements according to the user-preferred scale. For example, if the volume scale is large, the system will suggest requirements from systems suitable for large-scale projects, and for low-scale projects, it will generate requirements appropriate for smaller-scale systems. In addition, clustering techniques will be applied to group projects and features of the same type using different project and feature metrics.

8. CONCLUSION

This research paper proposes a framework that creates an automated system capable of generating requirements for any system or software using machine learning. Machine learning has become one of the essential components of computer science. Machine learning models are capable of learning, recognizing patterns, and making decisions with little or no human interaction. In principle, machines enhance accuracy and efficiency while considerably reducing the possibility of human error. For building the desired framework-learning model, which generates system requirements, initial user requirements are taken as input. From this information, the system can identify the features that the user needs and fill them accordingly. The framework primarily creates high-level descriptions that clearly define what the system will do and what it will not do. In the future, this framework can be enhanced by incorporating larger and more diverse datasets, applying advanced deep learning models for improved accuracy, and extending the similarity checks to include project descriptions and scalability factors. Additionally, clustering techniques may be used to group similar requirements, and integration with real-world industrial tools can transform the framework into a practical solution for software analysts.

9. REFERENCES

[1] V. Rooijen, B¨aumer, Platenius, Geierhos, Hamann, and Engels, From user demand to software service: Using machine learning to automate the requirements specification process, Jan. 1970. [Online]. Available: https:// ris.uni - paderborn.de/publication/97.

[2] K. Ryan, "The role of natural language in requirements engineering," in [1993] Proceedings of the IEEE International Symposium on Requirements Engineering, IEEE, 1993, pp. 240–242.

[3] H. Zhu and L. Jin, "Automating scenario-driven structured requirements engineering," in Proceedings 24th Annual International Computer Software and Applications Conference. COMPSAC2000, IEEE, 2000, pp. 311–316.

[4] A. Ning, H. Hou, Q. Hua, B. Yu, and K. Hao, "Requirements engineering processes improvement: A systematic view," in Software Process Workshop, Springer, 2005, pp. 151–163.

[5] L. Jiang, A. Eberlein, B. H. Far, and M. Mousavi, "A methodology for the selection of requirements engineering techniques," Software & Systems Modeling, vol. 7, no. 3, pp. 303–328, 2008.

[6] Z. J. Oster, G. R. Santhanam, and S. Basu, "Automating analysis of qualitative preferences in goal-oriented requirements engineering," in 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), IEEE, 2011, pp. 448–451.

[7] A. Chakraborty, M. K. Baowaly, A. Arefin, and A. N. Bahar, "The role of requirement engineering in software development life cycle," Journal of emerging trends in computing and information sciences, vol. 3, no. 5, pp. 723–729, 2012.

[8] J. H. Hayes, W. Li, and M. Rahimi, "Weka meets tracelab: Toward convenient classification: Machine learning for requirements engineering problems: A position paper," in 2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), IEEE, 2014, pp. 9–12.

[9] E. Parra, C. Dimou, J. Llorens, V. Moreno, and A. Fraga, "A methodology for the classification of quality of requirements using machine learning techniques," Information and Software Technology, vol. 67, pp. 180–195, 2015.

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

[11] F. Nazir, W. H. Butt, M. W. Anwar, and M. A. K. Khattak, "The applications of natural language processing (nlp) for software requirement engineering-a systematic literature review," in International conference on information science and applications, Springer, 2017, pp. 485–493.

[12] A. Rossanez et al., "Semi-automatic checklist-based quality assessment of natural language requirements= avalia¸c˜ao semi-autom´atica de qualidade de requisitos em lıngua natural baseada em checklist," 2017.

[13] F. Dalpiaz, A. Ferrari, X. Franch, and C. Palomares, "Natural language processing for requirements engineering: The best is yet to come," IEEE software, vol. 35, no. 5, pp. 115–119, 2018.

[14] T. Iqbal, P. Elahidoost, and L. Lucio, "A bird's eye view on requirements engineering and machine learning," in 2018 25th Asia-Pacific Software Engineering Conference (APSEC), IEEE, 2018, pp. 11–20.

[15] A. Ferrari and A. Esuli, "An nlp approach for cross-domain ambiguity detection in requirements engineering," Automated Software Engineering, vol. 26, no. 3, pp. 559–598, 2019.

[16] E. Dias Canedo and B. Cordeiro Mendes, "Software requirements classification using machine learning algorithms," Entropy, vol. 22, no. 9, p. 1057, 2020.

[17] S. Panichella and M. Ruiz, "Requirements-collector: Automating requirements specification from elicitation sessions and user feedback," in 2020 IEEE 28th International Requirements Engineering Conference (RE), IEEE, 2020, pp. 404– 407.

[18] N. Peinelt, D. Nguyen, and M. Liakata, "Tbert: Topic models and bert joining forces for semantic similarity detection," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7047– 7055.

[19] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro, "Natural language processing (nlp) for requirements engineering: A systematic mapping study," arXiv preprint arXiv:2004.01099, 2020.