AUTOMATING REQUIREMENTS ENGINEERING USING MACHINE LEARNING

by

Md.Mehedy Hasan Abid
17101033
Rubaya Neshat Tanna
17101204
Mahbubur Rahman Noyon
17101214
Jahidul Hasan Masud
17101418
Tahmina Akter
17301226

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

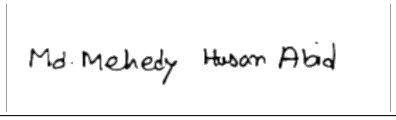
Department of Computer Science and Engineering
Brac University
September 2021

Declaration

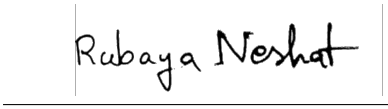
It is hereby declared that

1. The thesis submitted is our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

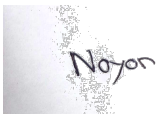
Student's Full Name & Signature:



Md.Mehedy Hasan Abid
17101033

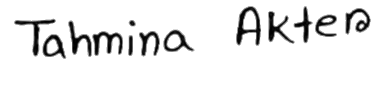


Rubaya Neshat Tanna
17101204



Mahbubur Rahman Noyon
17101214

Jahidul Hasan Masud
17101418



Tahmina Akter
17301226

Approval

The thesis titled "AUTOMATING REQUIREMENTS ENGINEERING USING MACHINE LEARNING" submitted by

1. Md. Mehedy Hasan Abid (17101033)
2. Rubaya Neshat Tanna (17101204)
3. Mahbubur Rahman Noyon (17101214)
4. Jahidul Hasan Masud (17101418)
5. Tahmina Akter (17301226)

As of summer, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 2 2021.

Examining Committee:

Supervisor:
(Member)

Mostafijur Rahman Akhond

Mostafijur Rahman Akhond
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department Of Computer Science And Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

The thesis is carried out in complete compliance with research ethics, policies, regulations and codes set by BRAC University. We have used various information from different sources in order to pursue the research. To collect data, we read articles, journals from different websites, etc. The sources we have used here are interpreted in our own terms and are properly mentioned as a reference. We appreciate and give credit to every source that helped us to continue our work. Lastly, we declare that five authors of this paper hold liability if any violation of BRAC University standard is found.

Abstract

Machine learning algorithms help to automate the process in many different problem domains. In the field of Software engineering. Requirement engineering is one of the first stages of software development. This research aims to automate the process of requirements engineering by integrating machine-learning algorithms, which should reduce the development cost and the possibility of human errors in several stages of the software engineering process. The thesis requires extensive machine learning algorithms to identify the best-suited technologies in the software engineering arena. Finally, we will identify some evaluation matrix to identify the effectiveness of our proposed algorithms for real-life software requirements specification.

Keywords: Machine Learning, Automate; Integrating, Extensive; Identify, Requirement engineering, Software engineering, NLP, BERT.

Dedication

We would like to dedicate our thesis to our parents who gave support by all means to come this far. Then to our respected supervisor Mostafijur Rahman Akhond sir.We could not have conducted our research without their instructions and guidelines. Lastly, with condolence, to all the people who had and still are suffering loss from data stealing and identity thieving issues.

Acknowledgement

In the name of Allah, Most Gracious, Most Merciful, Who has given us the strength and perseverance, we are grateful to our family members, as well as our supervisor, Mostafijur Rahman Akhond sir, for his consistent guidance and aid in envisioning that this research is possible. He helped us whenever we needed any sort of assistance and guided us to improve ourselves. For our parents' kind support and prayers, we are now on the verge of completing the final phase of our thesis. We also appreciate our fellow team members who held strong and united till the end to successfully complete the work. Finally, we want to thank our institution and its administration for providing us with a platform from which we may go one step closer to achieving our main objectives.

Table of Contents

List of Figures

Chapter 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 terms of processing the software requirements specification. Neural network works like human brain and nerve system . This technology find pattern in the data by itself. So, neural network can be used to find pattern 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 we see that 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 we can use automation, 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. We will try to Use machine-learning algorithms to build a framework that will generate software requirement specifications based on user requirements. To train the algorithm, we will collect existing SRS data from different sources. After that we will process those requirements by using NLP libraries and will use those data to train in a neural network.

In this paper, we first attempt to analyze a thorough study of the related works of our thesis. The literature review and background of NLP, Machine-learning algorithm as well as the BERT model we applied, are discussed in the next Chapter. The Dataset Analysis part, which includes data collecting, data cleaning, and data

organization, is covered in the next chapter. A detailed description of the dataset we collected using existing SRS data from different sources. After that we process those requirements by using NLP libraries and will use those data to generate requirements for the user given system. The subsequent chapter of Implementation describes the implementation procedure of our models. Finally, in the Conclusion section, we finalize our thesis and make recommendations for further research.

Chapter 2

Literature Review

2.1 Related Works

In this chapter of our report, we've discussed some of the previous research papers that we've looked into. Here are a few examples of prior projects:

The Authors Winkler, J., Vogelsang, A.[10] introduced a Natural language requirements specification 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 requirementsrelated 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 BERT-based 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, like 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.

Chapter 3

Proposed system Model

3.1 Problem Statement

To build our desired framework learning model that will generate the requirements for the system, at first, we need to take some initial requirements from the user. 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, then the user will be asked for the budget. Then, we will generate the system feature, requirements of functional and non-functional for the system from 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

Figure 3.1, depicts the workflow of how the learning algorithm is trained and generate the output from the existing data. To build the desired framework and learning algorithm, we needed a lot of software requirements specification data. Using this data, we would train our algorithm, and the more data we use to train our algorithm, the more it will give an accurate output. So, the data collection part was one of the main challenges to make the learning algorithm more rigid. As we collected data from different sources, the format of data was different from one another. So, we need to clean our data so that we can fit these data into our learning algorithm. After that, we organized the data into different groups for training purposes.

After working with data, we divide our data into two parts. A total of 80% of the random data was taken for training the learning algorithm, and the rest 20% data was taken for testing the learning algorithm.

In the learning phase, we need to do sentiment analysis, lemmatization, classification, and for that, we will use some natural language processing libraries. As we are working mainly with text data, it is convenient to use NLP libraries that work with text data. For that reason, the system will use BERT algorithm to make vectorize data as machine learning algorithms do not work with text data. After encoding text data to vectorize form-using BERT, we will use those data into cosine similarity to find similarity score. Finally, from similarity score of the features of the existing dataset we will generate some requirements for the user-preferred system.

After we finish creating the machine learning model and calculating similarity score, then we will ask the user for the preferred system information. Then, we will put that information in our machine-learning model, and then it will generate the recommended requirements for that system.

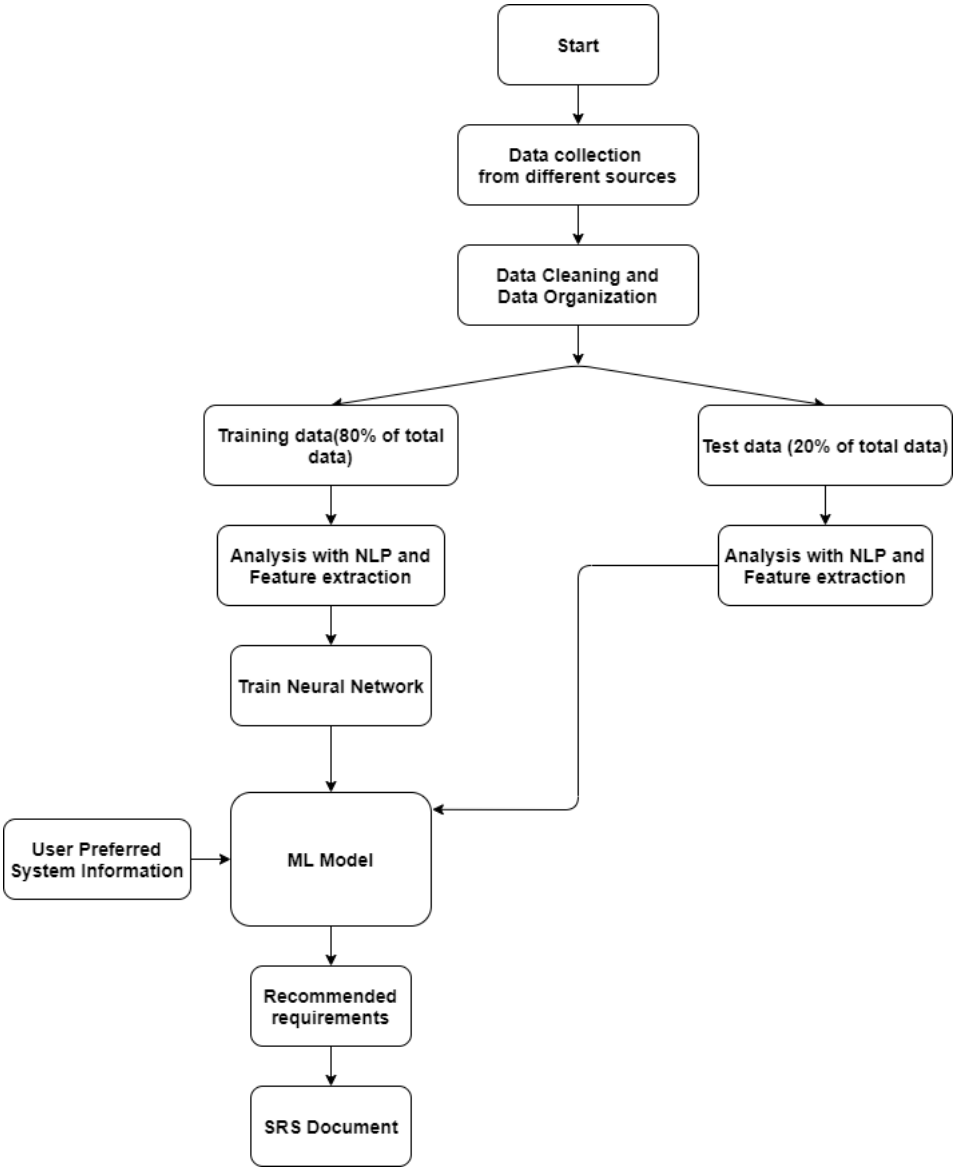


Figure 3.1: System Architecture Diagram

Chapter 4

Dataset Analysis

In this chapter, we have provided details of the data collection process, data cleaning and data 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, we have gone to several software companies to collect SRS (software requirements specification) data. We collected data through the responses to the questions that form the basis of understanding the problem or exploring our objective's idea. We also collect data from online open source. We used the Google search engine for searching available datasets on the internet. We have to go through some websites to collect software requirement data. Our institution's faculties also help us in collecting data. We faced some problems in collecting data, such as few companies not wanting to share their data said it is confidential. In addition, there are not many resources available on the internet regarding the SRS dataset.

4.2 Data Cleaning

The data set was collected from multiple sources. Some data were found on the internet, some came from the mail, and some were handwritten. So all datasets needed to merge in one format. After merging, the dataset contained some unnecessary data and a few incorrect data. Therefore, incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data was removed from the dataset. The dataset then contained only the format so that it can fit into the training input. If inaccurate data is not removed from the dataset, then outcomes and algorithms are unreliable, even though they may look correct.

4.3 Data Organization

After cleaning the dataset, we organize the database so that the data can be used more effectively. The dataset was created to learn the machine learning algorithms input, and the organization of the dataset was done accordingly. To make the training dataset, we made two datasets as the training dataset and labelled them as

Project dataset (Figure 4.1) and Feature dataset (Figure 4.2). Project dataset contains all the project names of the SRS document with a unique project Id of 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

Figure 4.1: Project Dataset

Feature dataset (Figure 4.2) contains project id from where those exact requirements are taken. This project ID is the same as the project dataset project ID.For each individual requirement, the feature dataset has its own Feature ID, name of that feature and description of that feature. Feature name describes a use case of the system. Feature Description elaborates process of the requirements functionality.

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 p
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 pa
1	6	Payment	Payment process will be fully verified through bank account details to detect 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 attempt is 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	Intilligent Search	The system will let seekers furthermore filter the search option by course type.
2	17	Intilligent 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 technician?s information.
3	20	Usability	Doctors uses the dashboard for patients 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 according to that.
3	24	Doctor	Doctors can download the report from the system whenever he wants.
3	25	Data integrity	Only 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 purpose.
3	27	Cross Platform Support	The system will operate on windows.
3	28	Performance	Database will be updated in real time.
3	29	Image	The system can import images
4	30	Identify hotspot	Able to identify the students? smartphones using 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 attendence	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 a time i.e. allow 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.

Figure 4.2: Feature Dataset

4.4 Entity Relationship Diagram

Basically, there will be three entities in our system (Figure 4.3). First One will be project. Project's primary key will be project ID. Then we have another entity is feature. In feature entity, Feature ID will be primary key and project ID will be foreign key and it will be taken from project ID. Lastly, we have another entity is Similarity Check. In similarity check, Similarity_score_ID will be primary key and

Project_ID1, Feature_ID1 will be foreign key. This will be generated through the implementing algorithm from project and feature. One project can have one or many features.

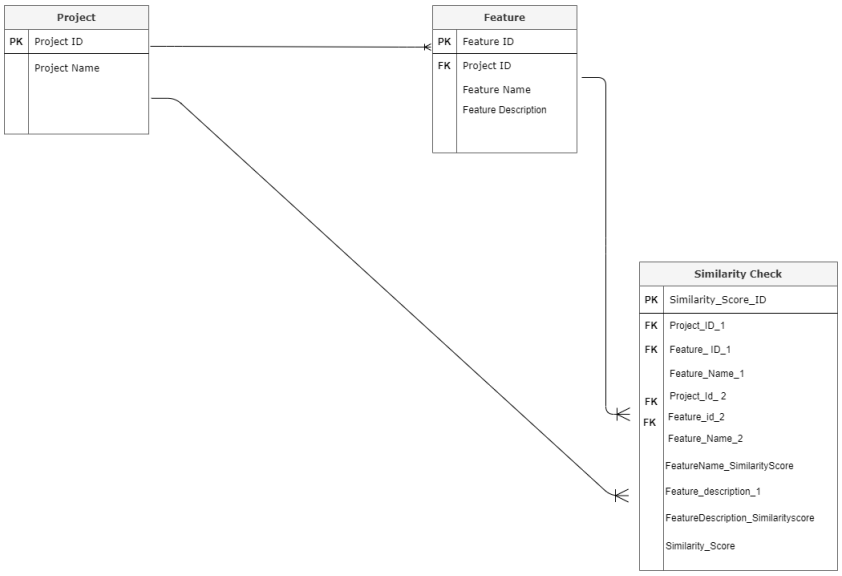


Figure 4.3: Entity Relationship Diagram

Chapter 5

Model Specification

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.

three key reasons why BERT is so excellent in my opinion. No. 1: Pre-trained on a large amount of data. No. 2: Take into account the context of a word. No. 3: Opensource software.

In our paper, we have collected data (such as feature name, feature description etc.) from lots of research papers. By using BERT, we have compared the similarity of one document to another document. The similarity score shows if the meanings of two texts are similar or different. To calculate the similarities, we have to

use semantic similarity and cosine similarity sklearn. The similarity between texts or documents is scored using a defined metric in semantic similarity. And Cosine similarity is a measurement that assesses how similar papers are independent of size.

Semantic similarity, this attribute scores words depending on how similar they are, even if they are not exact matches. It utilizes Natural Language Processing (NLP) techniques such as word embedding.

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.

$$similarity(A, B) = \frac{A * B}{||A|| * ||B||} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5.1}$$

Chapter 6

Requirement Generation

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)))
```

Figure 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,

```
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]
)
```

Figure 6.2: Algorithm (Continued)

We calculate every individual project feature name with another project feature names. At first, the names are passed to the BERT semantic similarity model, which will encode and vectorize the text data so that the data can be ready to apply for calculation on different machine learning algorithms. After that, this vectorize data will be passed to the cosine similarity function which will generate the similarity score. Each of the feature name of a project will have a similarity score with other project features name. After that, with same procedure of the feature name similarity score calculation, feature description similarity will be measured amongst the individual project feature description each other of the different projects.

After getting these two scores, we will calculate the final similarity score against the two requirements of different projects.

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 Rules	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 other ? contents they want.	0.555434585	0.577331781
2	1	1	Registration and Login	2	12	Instructor Rules	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 of your own course.	0.413554370	0.492203653
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	0.588112295	All the users need to get registered and then login in order to bid.	The courses will also include online exams.	0.466497272	0.515143281
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.	0.530310750	0.556750727
6	1	1	Registration and Login	2	16	Intilligent 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	Intilligent Search	0.713651538	All the users need to get registered and then login in order to bid.	Intelligent system to show course suggestions.	0.385803461	0.516942692
8	1	1	Registration and Login	2	18	Availability	0.588888288	All the users need to get registered and then login in order to bid.	Our system should be available 24/7.	0.456304729	0.509338152

Figure 6.3: Similarity Check Dataset

Once similarity check dataset calculation is ready, this system can take user input of the system name from the user. Then, the user given system name will be performed semantic similarity check with each project name of the project dataset. If the score between the user given system and other project matches the threshold limit condition then the project will be taken for the future calculation. After getting all the projects that meets the threshold limit condition with the user given system name, the requirements will be generated form the similarity scores of each projects features similarity scores with the other projects that matched the threshold.

6.2 Experimental Result and Analysis

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

Figure 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 existing project's requirements. Each feature contains a description according to that feature, so that, user can understand the requirements technical aspect. Also, as this system generates output based on the existing system, so they can get idea of the other systems feature related to the user given system and use those requirements.

```
Suggested Requirments for   online captain voting system


Feature Name :  Security ----->>  Feature Description :  The password saved in the database should be hashed. The whole system would carry out an anonymously.
Feature Name :  Multiple Access  ----->>  Feature Description :  Allow multiple accesses simultaneously.
Feature Name :  Cross platform support ----->>  Feature Description :  The system should be able to work on any Web browser
Feature Name :  Admin ----->>  Feature Description :  Only admin can see personal record of voters and candidates
Feature Name :  Registration  ----->>  Feature Description :   Register with all the valid information
Feature Name :  Login ----->>  Feature Description :   Login with Student id and password
Feature Name :  User ----->>  Feature Description :  Users request for unique id generated by admin
Feature Name :  Search Election ----->>  Feature Description :  Search ongoing elections
Feature Name :  Look and feel ----->>  Feature Description :  The system should be user friendly
Feature Name :  Storage ----->>  Feature Description :  The database should have enough data storage
Feature Name :  Voting Criteria ----->>  Feature Description :  Only a user can cast vote?
```

Figure 6.5: Requirement Generation

Chapter 7

7.1 Future Perspectives

The algorithm is still in its development stage. So far, we are checking the similarity check with the project name but in future we will also match the similarity check with the project description and also look at the project volume. By observing the project volume, we will generate requirements according to the user-preferred scale. For example, If the volume scale is large, we will suggest requirements from the system that for large scale and for low scale we will generate low scale projects requirement from our system. In addition, we will use some clustering techniques to cluster projects and features of the same type using different project and feature metrics.

7.2 Conclusion

This paper proposed a framework that creates an automated system that can generate automated requirements for any system or software by using machine learning. Machine learning has been one of the essential parts 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 reducing (or considerably reducing) the possibility of human mistake. For building our desired framework-learning model, which generates the system's requirements, taking initial user requirements, and from this information, we will be able to know the features that the user wants, and we can fill them. We will mainly create high-level descriptions that can clearly describe what the system will do and what it will not do.

Bibliography

[1] v. Rooijen, Bäumer, 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= avaliação semi-automática 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.