IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 August 2024, accepted 25 September 2024, date of publication 7 October 2024, date of current version 30 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3475293

RESEARCH ARTICLE

A Systematic Review of AI-Enabled Frameworks in Requirements Elicitation

VAISHALI SIDDESHWAR, (Graduate Student Member, IEEE), SANAA ALWIDIAN, AND MASOUD MAKREHCHI, (Member, IEEE)
Department of Electrical, Computer and Software Engineering, Ontario Tech University, Oshawa, ON L1G 0C5, Canada
Corresponding author: Vaishali Siddeshwar (vaishali.siddeshwar@ontariotechu.net)

ABSTRACT Employing Artificial Intelligence techniques to address challenges in requirements elicitation is gaining traction. Although nine systematic literature reviews have been published on AI-based solutions in the requirements elicitation domain, to our knowledge, these studies do not cover a broad spectrum of elicitation tasks, data sources used for training, the performance of these algorithms, nor do they pinpoint the strengths and limitations of the algorithms used. This study contributes to the field by presenting a systematic literature review that explores the use of machine learning and NLP techniques in the elicitation phase of requirements engineering. The following research questions are addressed: 1) What elicitation tasks are supported by AI and what AI algorithms were employed? 2) What data sources have been used to construct AI-based solutions? 3) What performance outcomes were achieved? 4) What are the strengths and limitations of the current AI methods? Initially, 665 papers were retrieved from six data sources, and ultimately, 122 articles were selected for the review. This literature review identifies fifteen elicitation tasks currently supported by artificial intelligence and presents twelve publicly available data sources used for training these approaches. Furthermore, the study uncovers common limitations in current studies and suggests potential research directions. Overall, this systematic literature review provides insights into future research prospects for applying AI techniques to problems in the requirements elicitation domain.

INDEX TERMS Requirement engineering, requirement elicitation, machine learning, deep learning, natural language processing, neural networks, large language models.

I. INTRODUCTION

Requirements Engineering (RE) is the first step in software development life cycle. The objective of this stage is to model user demands and also ensure that the system to be developed operates as per the expectations of the customer. The RE phase begins with communication between the requirements engineer and the stakeholders of the envisioned system. There are five steps that correspond to the activities conducted to document user needs as precisely as possible. These activities include- elicitation, analysis, specification, verification and management.

In the context of software development process, it is the *requirement elicitation* phase which is the foremost process conducted, determines the successful development of a software system. In conventional requirements engineering process, requirements are elicited in natural language as plain text from data collection methods such as interviews, workshops, and brainstorming sessions. The process of requirements elicitation is challenging, often hindered by ambiguous stakeholder needs, conflicting priorities, and the dynamic nature of requirements. Traditional methods frequently struggle to effectively capture and consolidate diverse perspectives, leading to misunderstandings and project delays. Moreover, Textual requirements often lack the specificity needed by technical teams to effectively progress through the software development lifecycle. Extracting actionable technical details from these high-level descriptions is crucial but time-consuming and prone to errors when done manually.

Artificial Intelligence(AI)-enabled solutions offer a potential solution to this problem by automating data analysis,

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Ni.

pattern recognition, and applying natural language processing (NLP) methods to automatically extract useful components from raw textual customer needs. This enables the identification of hidden dependencies, inconsistencies, and valuable insights from vast datasets.

To the best of our knowledge, nine literature reviews have been published between January 2012 and August 2023 that touch upon the applications of AI in activities related to Requirements Elicitation. A review by Meth et al. [1] discussed strategies for evaluating approaches but omitted any mention of the performance achieved by these methodologies in the requirements elicitation domain. Five studies confine their review to specific aspects of the elicitation activity. Perez-Verdejo et al. [2] and Lopez et al. [3] focus only on the classification of software requirements. Ahmad et al. [4] focuses on the identification and classification of software requirements on Stack Overflow. Reviews from Binkhonain and Zhao [5] and Handa et al. [6] focus on identifying and classifying non-functional requirements. These studies overlook other crucial elicitation activities and do not cover the peak performance levels achieved by the approaches. Reviews conducted by Lim et al. [7] and Sampada et al. [8] present the data sources used for elicitation purposes. However, these studies do not uncover the highest performance benchmarks attained by the AI approaches. A study by Cheligeer et al. [9] uncovers the data sources generally used in the current literature but does not review the optimal AI models for solving elicitation tasks, the pre-processing techniques applicable to specific elicitation tasks, nor does it provide information regarding the maximum performance achieved by these models.

Despite that NLP and other AI-based solutions have been deployed in several software and requirements engineering areas, to the best of our knowledge, there is no study available that summarizes, in a comprehensive manner, not only the applications of AI in the requirements elicitation domain, but also the performance of AI algorithms, the data sources used to train the AI models and the disadvantages of the present AI-based solutions for solving the problems in requirements elicitation.

This paper aims to examine the application of numerous AI techniques that not only includes Machine Learning, but also Neural Networks, Deep Learning and Natural Language Processing techniques for the problems related to requirements elicitation. This study focuses also on technical details, such as the performance of AI algorithms and the data sources used to train the AI models featured in the selected studies.

The main contributions of this study are as follows:

- Provide an overview of AI algorithms that are currently used to automate common challenges in requirements elicitation phase, along with analyzing the performance of these AI algorithms and the data sources used to train the AI models.
- Identify elicitation activities enhanced by AI applications.
- Identify the limitations of current AI methods.
- Identify research gaps and propose novel research opportunities

The remainder of the paper is structured as follows: In Section II related review papers in the field of RE using AI methodologies are summarised. Section III presents the research methodology used in the systematic review. Section IV discusses the results of the review. Section V recommends future directions of research. Finally, conclusion is provided in Section VI.

II. RELATED WORKS

In this section, we present a few existing SLRs that are centered on the Artificial Intelligence methodologies in eliciting requirements. To our knowledge, 9 literature reviews have been published between January 2012 to August 2023 that touch upon the applications of AI in activities related to Requirements Elicitation. The details pertaining to these review articles are shown in Table 2.

Meth et al. [1] conducted a comprehensive review emphasizing the degree of automation within the requirements elicitation domain. Their investigation categorized existing studies into two main streams: semi-automated and fully automated elicitation methodologies. Moreover, the study identified four key elicitation activities: Abstraction identification, Requirement model generation, Requirement quality analysis, and Requirements identification. Additionally, it was noted that the most common evaluation strategies including simulations, case studies, proofs-of-concept, or combinations thereof. Their study did not encompass an examination of AI techniques integral to automated elicitation activities, nor did it delve into the data sources utilized for training their solutions. While they discussed strategies for evaluating approaches, they omitted any mention of the performance metrics achieved by these methodologies.

The main theme of review by Ahmad et al. [4] was the effective identification and classification of the software requirements on Stack Overflow (SO) for building quality systems. Towards this, they categorized the different types of machine learning techniques and algorithms used in identifying the software requirements on Stack Overflow data. They also identified that precision and recall are amongst the most commonly utilized evaluation methods for measuring the performance of these ML algorithms. Their study is confined to utilizing textual data exclusively sourced from Stack Overflow and neglects to identify the essential data sources required for constructing these AI solutions. Furthermore, their discussion solely revolves around methods for evaluating AI-based solutions, overlooking the critical examination of the performance levels attained by existing solutions.

Perez-Verdejo et al. [2] provide insights into the utilization of machine learning-based solutions specifically aimed at the classification of software requirements. They highlighted

five commonly employed training data sources crucial for training the machine learning models under consideration. In a complementary manner, the reviews conducted by Binkhonain and Zhao [5] and Handa et al. [6] delve into AI-based techniques tailored for identifying and classifying non-functional requirements. All three studies underscore the prevalence of Precision and Recall as the primary metrics used to evaluate the performance of these approaches. However, their analyses are predominantly confined to this specific aspect of requirements elicitation, overlooking other crucial elicitation activities. Additionally, these studies omit pivotal details regarding the optimal AI algorithms utilized in existing techniques and the peak performance levels achieved by these algorithms.

A systematic review by Lim et al. [7] presented the current state-of-art approaches for data-driven requirement elicitation where they analyzed the data sources, data processing methods, and learning techniques used for the purpose of elicitation. Their research uncovered that automated requirements elicitation relies on the application of supervised machine learning methodologies leveraging data sourced from dynamic repositories such as software repositories, online reviews, blogs, and discussions. However, their work does not mention the highest performance benchmarks attained by these algorithms.

In their study, Lopez et al. [3] identified sixteen techniques centered on Artificial Neural Networks for requirement classification, alongside detailing the data sources utilized in these studies. Nevertheless, their review overlooks the crucial text pre-processing steps that significantly impact the accuracy of AI algorithms. Furthermore, it predominantly concentrates on a narrow aspect of elicitation, specifically the classification of requirements, without delving into other facets. Additionally, the study fails to address the highest performance benchmarks achieved by these algorithms.

Sampada et al. [8] provided the state-of-the-art review of techniques to automate the requirement elicitation processes. But they have not identified distinct activities that have been automated using AI techniques. Moreover, the study does not specify the underlying data sources used for achieving state-of-the-art performance, and also the highest performance benchmarks attained by these algorithms.

Among the existing studies, one review published by Cheligeer et al. [9] is closely related to this study. Their work conducted a review aiming to offer insights into the application of Machine Learning techniques for requirements elicitation tasks. While both studies aim to identify various data sources for constructing data-driven AI models, Cheligeer et al.'s [9] work lacks a numerical comparison of different state-of-the-art models. Furthermore, it does not review the optimal models and pre-processing techniques applicable to specific elicitation tasks, nor does it provide information regarding the maximum performance achieved by these models.

Our study tries to address the aforementioned gaps through examining the application of diverse AI techniques, including Machine Learning, Neural Networks, Deep Learning, and Natural Language Processing, to address challenges inherent in requirements elicitation. A primary focus is on the technical aspects of AI algorithms and the associated datasets employed in relevant studies.

By systematically analyzing existing research, this study seeks to contribute to a deeper understanding of emerging AI techniques for automating various requirements elicitation tasks, thereby providing valuable insights for both practitioners and researchers in the field.

III. RESEARCH METHODOLOGY

This systematic review is conducted following the principles of Kitchenham [10]. The following subsections discuss the steps conducted in this literature review, namely, planning, and executing the review.

A. REVIEW PLANNING PHASE

This subsection describes the first phase of the systematic literature review i.e. the planning phase. In this phase the research questions, search strategy and inclusion criteria are established.

1) RESEARCH QUESTIONS

The goal of this literature review is to identify the trends in AI-based solutions in requirements elicitation activities. Following are the research questions that are formulated based on the research objective of the review:

- **RQ1**: Which elicitation tasks are currently supported by AI and what AI algorithms were employed?
- **RQ2**: What data sources are used for model training?
- **RQ3**: What performance outcomes were achieved?
- **RQ4**: What are the strengths and limitations of the current AI methods?

2) SEARCH STRATEGY

After defining the research questions, the search strategy has to be built to gather the available research articles for data synthesis. Two strategies were used to search potential studies. First, a manual search in International Conference on Requirement Engineering and International Conference on Software Engineering was performed. As a second search strategy, a search string that includes the main keywords related to the problem was crafted. The string consisted of three parts, namely P1, P2, and P3. P1 included the keyword 'requirement elicitation' OR 'elicitation*' to retrieve papers that are relevant to elicitation engineering domain.

P2 comprised of keywords related to Artificial Intelligence techniques, such as 'AI'', 'Artificial Intelligence', 'Machine Learning', 'Deep Learning' OR 'NLP', 'Neural Networks'', 'NN', 'Language Model*', and 'Natural Language Processing'. To include materials that contained gaps in the AI-based solutions, synonyms of the keyword 'drawback' and its synonyms, such as 'limitations', 'shortcomings' was added

Review Title	Year Published	Reference	Number of Studies Reviewed	Publication Years Examined	Review Scope
Machine learning in requirements elicitation: a literature review	2022	[9]	86	2007 to 2021	Requirement elicitation activities that are supported by ML; the data sources used to train ML algorithms; available technologies, algorithms, and tools are used to build ML-based requirement elicitation
A Systematic Literature Review on Using Machine Learning Algorithms for Software Requirements Identification on Stack Overflow	2020	[4]	12	2011 to 2019	ML-based techniques and algorithms used for identifying software requirements Stack Overflow.
Data-Driven Requirements Elicitation: A Systematic Literature Review	2021	[7]	68	2009 to 2020	Identifies data used for data-driven Requirement Elicitation process; methods and techniques used for processing the data
A Systematic Literature Review on Machine Learning for Automated Requirements Classification	2020	[2]	13	2010 to 2019	Summarizes the ML algorithms adopted for classifying requirements. What are the categories of requirements identified; summarizes the performance metrics of the algorithms.
Automatic Classification of Software Requirements using Artificial Neural Networks: A Systematic Literature Review	2021	[3]	14	2016 to 2020	types of Neural Networks used to classify Non-Functional Requirements; ML-based techniques used for identifying NFR; What are the categories of requirements identified; summarizes the performance metrics used to measure the performance of the algorithms.
A review of machine learning algorithms for identification and classification of non-functional requirements	2019	[5]	24	2007 to 2017	Reports the most commonly used ML-based approaches for identification and classification of NFRs; find the most used metrics to measure the performance of these approaches.
The state of the art in automated requirements elicitation	2013	[1]	36	1992 to 2012	Categorize the studies in the area of automated requirements elicitation;
A Review on Advanced Techniques of Requirement Elicitation and Specification in Software Development Stages	2020	[8]	13	2011 to 2019	Identifies ML Algorithm or model used for requirements elicitation
An Inclusive Study of Several Machine Learning Based Non-functional Requirements Prediction Techniques	2020	[6]	8	2010 to 2017	Identifies ML Algorithms used to classify the NFRs into different categories; the review also mentions the accuracy achieved.
This Work			122	2013 to 2023	The study unveils 15 elicitation tasks currently automated through AI solutions and extracts the state-of-the-art AI algorithms, along with their performance metrics on publicly available data sources.

in P3. The search string was tailored manually for each data source before execution.

3) INCLUSION CRITERIA

The search string developed in the previous section is broad, and not all studies are pertinent for this literature review. Therefore, inclusion criteria were defined to shortlist the most relevant studies in accordance with the research objective. The Table 1 summarizes the inclusion criteria. The first and third criteria lists the data sources and the category of the papers. The second criterion ensures that the studies are recent and are written in English. Lastly, the fourth criterion specifies that the keywords are searched within the titles and abstracts of the studies

B. CONDUCTING RESEARCH PHASE

Our goal was to cover a broad spectrum of search processes to identify the maximum number of literature relevant to the requirements elicitation domain. To achieve this objective, a three-phased process was implemented. An overview of paper selection process is shown in Figure 1.

In the first stage, 'Search Stage 1,' we executed the constructed search string in selected digital libraries, including ACM, IEEE, Scopus, Science Direct, and Springer databases. Additionally, we manually gathered papers listed in the

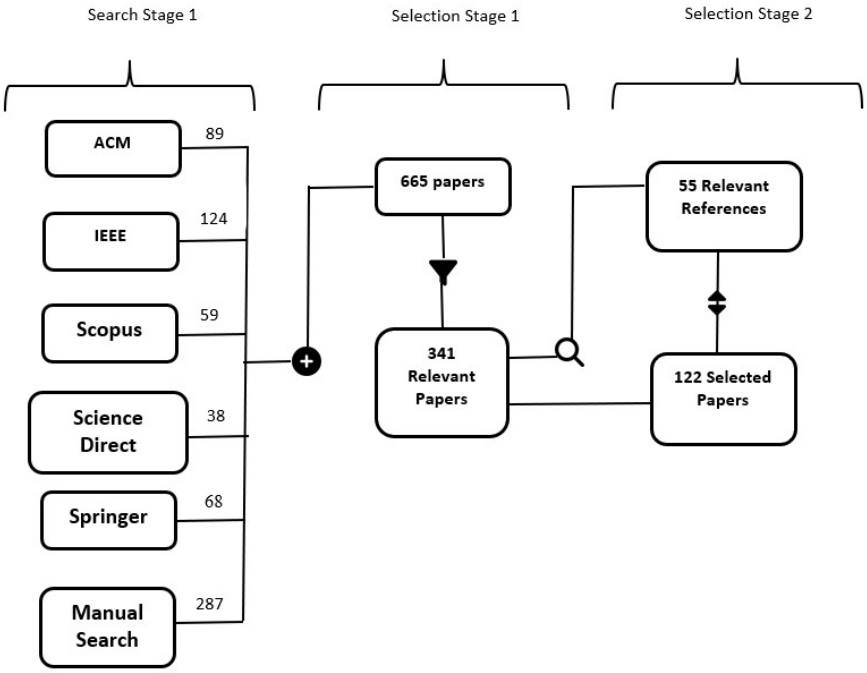

FIGURE 1. An overview of study selection process.

TABLE 1. Inclusion criteria for study selection.

Criteria	Description
Data Sources	ACM
	IEEE
	ScienceDirect
	Scopus
	Springer
	AIRE
	IREC
	ICSE
Publication Year	2012 to 2023
Article type	journals, conference and workshops
filter applied on	title and abstract

AIRE, IREC, and ICSE workshops and conferences. This phase returned 665 papers. However, many duplicate titles were found because IEEEXplore contains papers from both SCOPUS and ACM, and some papers manually searched from AIRE, IREC, and ICSE websites were also present in IEEEXplore.

Therefore, in the second stage, 'Selection Stage 1,' we removed duplicate titles from the articles, reducing our search space to 341 papers. We then filtered through these by reading the titles and abstracts, removing non-relevant papers. For the final selection process, we analyzed the studies based on four quality questions. This analysis was conducted manually; if most of the questions were answered with a 'yes' the study was selected for review. The list of questions is as follows:

- Q1. Does this study clearly outline the objective?
- Q2. Was the evaluation method properly stated?
- Q3. Was the training dataset described?
- Q4. Are all the study research questions discussed?

IV. RESULTS OF THE REVIEW

Following the process described in section III, the selected primary studies were reviewed and analyzed carefully to find the answers to the research questions. Figure 2 illustrates the cumulative growth in the number of relevant publications over the past decade. Notably, the figure reveals a significant surge in AI-driven elicitation task research post-2020, with the year 2020 marking the peak at 24 published papers. The following subsections provide an in-depth discussion of the findings related to the four research questions. Subsection IV-A explores the elicitation tasks supported by AI techniques and the various algorithms employed (RQ1). Subsections IV-B and IV-C delve into the data sources used for training these AI-based algorithms (RQ2) and the highest performance levels they achieved (RQ3). Finally, Subsection IV-D examines the strengths and limitations of the AI methods (RQ4).

A. RQ1: ELICITATION TASKS AND AI TECHNIQUES EMPLOYED

This section begins by discussing the most commonly addressed elicitation tasks and then presents the AI techniques employed to tackle these tasks. Figure 3 illustrates the top 10 elicitation activities most frequently automated with AI, while the list of all elicitation tasks that are automated using AI is shown in Figure 5. Figure 3 reveals that the majority of studies focus on mining requirements from reviews, as seen in studies by Iqbal et al. [11], and Khan et al. [12], which explore the discovery of requirements through feedback expressed on social media

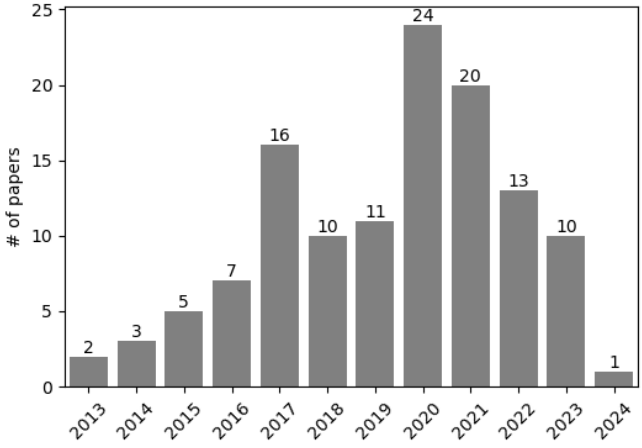

FIGURE 2. Papers with AI-based solutions published between 2012 to 2023.

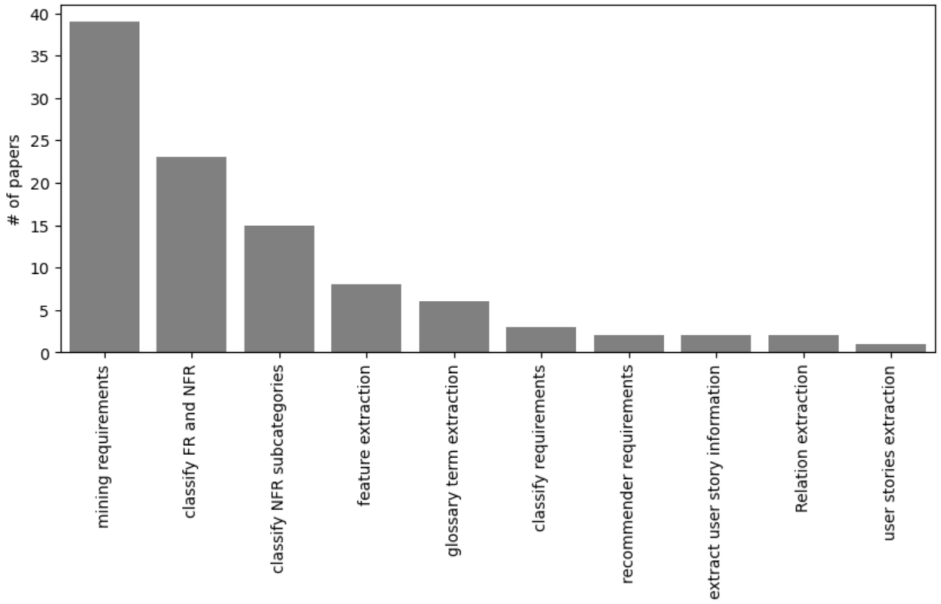

FIGURE 3. Distribution of elicitation tasks currently supported by AI.

platforms, including app stores and user forums. Another frequently addressed task is classifying existing requirements into Functional (FR) and Non-Functional Requirements (NFR), as demonstrated in Hey et al. [13], where existing software requirements are utilized. Additionally, categorizing non-functional requirement statements into specific NFR subcategories, such as, in Munaiah et al. [14], represents another common elicitation task. Researchers have also proposed methods to automatically create glossaries containing technical terms, synonyms, and domain concepts present in requirements documents. These approaches typically combine linguistic and statistical techniques. In contrast to these well-explored tasks, there is significantly less literature focused on other tasks, such as extracting user stories from natural language requirements in Siahaan et al. [15], mapping relationships between requirements in [16], and identifying components within user stories Raharjana et al. [17], with only one study found for each of these topics. A summary of all the elicitation tasks that are automated using AI is shown in Figure 5; it reveals that following elicitation tasks are automated using AI - Mining requirements, classify Functional and Non-Functional requirements, classify NFR subcategories, ambiguity detection, glossary extraction, user story extraction, detect privacy requirements, categorizing requirements into topics, identifying uncertain requirements, transforming requirements to specification, relation extraction, detect inconsistent and redundant requirements, detect privacy requirements, extract actions and actors from requirements.

The discussion now shifts to the AI techniques utilized in addressing the aforementioned elicitation tasks. In the domain of requirements elicitation, studies rely on textual

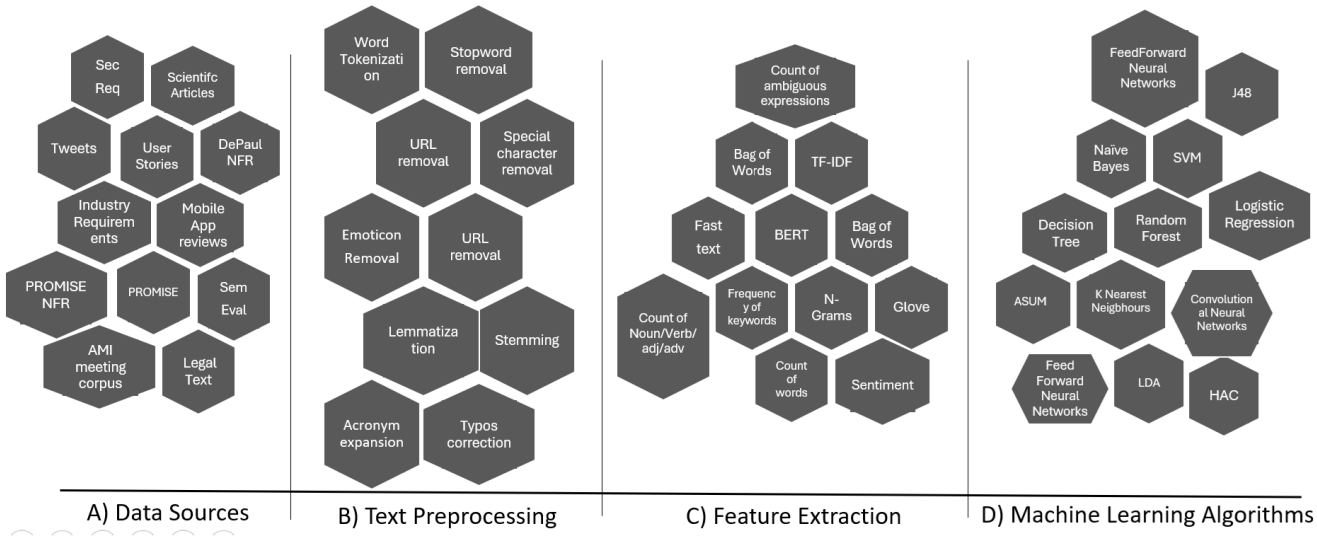

FIGURE 4. An overview of AI algorithms.

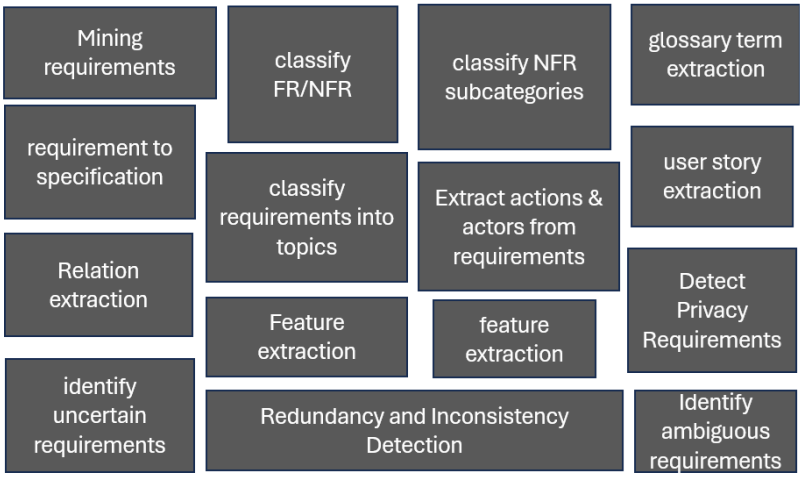

FIGURE 5. Elicitation tasks from selected studies.

requirements as their input, whether in a structured format, as user stories or unstructured format, as natural language requirements. The studies then apply NLP techniques to distill meaningful features from the data. A typical NLP pipeline consists of various components that are sequentially applied to process and analyze natural language text. These components work in tandem to transform raw text data into structured information. While the number of components may vary depending on the specific task, the most commonly found components in NLP pipelines are depicted in Figure 4(A). A complete list of NLP techniques and AI algorithms used in the analyzed studies are shown in Appendix Table 4.

The content in the following subsections is organized as follows: subsubsection IV-A1 explores diverse methods for text processing, while subsubsection IV-A2 presents text vectorization approaches that are helpful for extracting features from the processed text and representing them as numerical vectors. Finally, subsubsection IV-A3 discusses both supervised and unsupervised machine learning algorithms.

1) TEXT PRE-PROCESSING TECHNIQUES

Text preprocessing refers to the process of cleaning and preparing textual data before it is passed to a NLP pipeline for further analysis. It involves several steps to transform raw text into a format that is suitable for computational processing and analysis. The most commonly used preprocessing steps, as shown in Figure 4(B), are mentioned below. Most of the studies describe at least one text-preprocessing methodology.

a: TOKENIZATION

Textual requirements often consist of multiple words or phrases that convey specific aspects or functionalities.

Tokenization decomposes the requirements into a list of elements such as words, symbols, and punctuation, that are also called as tokens, providing a more granular representation of the requirements.

b: TEXT DEDUPLICATION

This is the process of identifying and removing duplicate or near-duplicate instances of text within a dataset.

Expand contraction is an important preprocessing step in NLP, especially when dealing with informal or colloquial text, as contractions are commonly used in casual language. Contractions are shortened forms of words or phrases that are created by combining two words and replacing one or more letters with an apostrophe. This step converts contracted words into their full forms in text data.

Expanding contractions involves replacing these contracted forms with their full equivalents. For example, ''can't'' is a contraction of ''cannot'', ''won't'' is a contraction of ''will not'', and ''I'm'' is a contraction of ''I am''. This using automated text processing techniques. By expanding contractions, the text becomes clearer and more understandable, making it easier for downstream NLP tasks such as text classification to accurately interpret the text.

c: STEMMING

Stemming is a text normalization technique used in NLP to reduce words to their root form, called the 'stem'. The goal of stemming is to simplify words by stripping them of affixes such as prefixes and suffixes, while still retaining the core meaning of the word. For example, after stemming, word ''analyzing'', ''analyze,'' and ''analyzed'' would be reduced to the common stem ''analyz''. Stemming algorithms typically use heuristic rules to remove affixes from words, without considering the context or meaning of the word. This means that stemming may sometimes produce stems that are not actual words, but rather word fragments. As a result, more advanced techniques like lemmatization, which consider the linguistic context of words and produce valid dictionary forms (lemmas), are preferred in certain applications.

d: LEMMATIZATION

Lemmatization is a text normalization technique used in NLP to reduce words to their base or dictionary form, known as the ''lemma''. Unlike stemming, which simply removes affixes from words to derive a common root form, lemmatization considers the context and meaning of words, ensuring that the resulting lemma is a valid word found in a dictionary.

For example, the word ''analyzing'' would be lemmatized to ''analyze''. Lemmatization takes into account the part of speech (POS) of each word, ensuring that the lemma corresponds to the appropriate form of the word for its context.

Lemmatization typically involves the use of lexical resources like dictionaries or morphological analysis to map words to their lemma forms. This makes lemmatization more linguistically accurate than stemming, as it produces valid dictionary forms of words rather than arbitrary word fragments. Lemmatization is particularly useful in NLP tasks where word accuracy and interpretability are important, such as text classification,

e: STOPWORDS REMOVAL

Stopwords removal is a preprocessing step in NLP aimed at eliminating common words that do not carry significant meaning in the context of the analysis. Stopwords are frequently occurring words that occur across multiple documents and typically do not contribute much to the overall understanding or interpretation of the text.

Examples of stopwords include common words such as ''the'', ''is'', ''and'', ''to'', ''in'', ''of'', ''a'', ''an'', ''for'', etc. These words are so ubiquitous in the language that they often add noise to the analysis rather than useful information. A higher frequency of occurrences of stop words adversely effects on the model training process. By removing stopwords from textual data, the focus shifts to more meaningful words, allowing for more accurate and efficient analysis.

Stopwords removal is commonly performed as part of text preprocessing in various NLP tasks, such as, requirements classification, and topic modeling. It is important to note that stopwords removal should be done with caution, as some stopwords may carry importance in certain contexts. Therefore, for tasks such as sentiment analysis, it is essential to carefully consider the impact of stopwords removal on the analysis and adjust the list of stopwords accordingly.

f: LOWERCASING

Converting all text to lowercase to ensure consistency and prevent duplication of words based on case variations.

g: NOISE REMOVAL

Eliminating irrelevant characters, symbols, or formatting from the text, such as punctuation marks, special characters, HTML tags, etc., which are not relevant to the analysis.

h: PARTS OF SPEECH (POS) TAGGING

POS tagging plays a crucial role in many NLP tasks by providing valuable information about the grammatical structure of text, which can be used to enhance the performance of various language processing algorithms. The main goal of POS tagging is to analyze the syntactic structure of a sentence by categorizing each word according to its grammatical function within the sentence. This information is crucial for understanding the meaning and structure of text. POS Tagging involves assigning grammatical tags, known as parts of speech, to words in a text corpus. The parts of speech include categories such as nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, and interjections, among others.

POS tagger is used in approaches in Rahmi et al. [18] to extract requirement-related information from natural language requirements. Their approach builds a

software-specific vocabulary by looking for the common tokens that appear in both online news corpus and software-specific corpus. After this, requirement-related information is extracted by finding sentences with Verb + Noun POS tags. Similarly, POS tagging is also used in studies([15], [17], [17], [19], [20], [21], [21], [22], [23], [24], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48]). All these studies utilize rule-based POS taggers such as- StanfordNLP, Spacy and NLTK (Natural Language Toolkit).

Named Entity Recognizer (NER) is an NLP algorithm that identifies and classifies named entities in text into pre-defined categories such as names of persons, organizations, locations, dates, numerical expressions, and more. Named entities are specific entities mentioned in text that have real-world significance and can be categorized into specific types. For example, in the sentence ''Apple is headquartered in Cupertino, California'', ''Apple'' is a named entity of type organization, and ''Cupertino, California'' is a named entity of type location. NER is an important component in many NLP applications, particularly in information retrieval and knowledge extraction tasks, where identifying and categorizing named entities can help in extracting structured information from unstructured text data. Named Entity Recognizer (NER) from spaCy is used in [15], [17], [32], and [47], Stanford NER is used in [39], and NLTK's NE Chunker is used in [49] for identifying named entities which represent an organization.

i: CHUNKING

also known as shallow parsing, used in studies ([15], [31], [34], [50], [51], [52]) is the process of detecting syntactic constituents such as Noun Phrase and Verb Phrase in a sentence. The chunking process decomposes a sentence into noun phrase (NPs), verb phrases (VPs), and prepositional phrases (PPs). The first module in the pipeline of text chunker is *Tokenizer*, which breaks the input text into tokens. The next module is a *Sentence Splitter*, that returns sentences of a text. Subsequently, the *POS Tagger* annotates the tokens with POS (part-of-speech) tag such as Noun, Pronoun, Adjective and Verb. Further, the *Name Entity Recognizer* identifies entities in the text, as used by Arora et al. [26]. Chunking is used in Gemkow et al. [43]. The authors use treebank_chunk from nltk.corpus library which uses a statistical approach to chunking and is trained with the Treebank corpus data for which the correct parse tree has been determined by human experts.

j: DEPENDENCY PARSER

is a natural language processing algorithm that analyzes the grammatical structure of a sentence by identifying the dependencies between words. It represents these dependencies as directed arcs or edges between words in a dependency tree, where each word is a node and the arcs indicate the relationships between them. The dependencies captured by a dependency parser typically represent syntactic relationships such as subject-verb, object-verb, modifier-head, and so on. The research conducted by Wang et al. [25] adopts Stanford Dependency Parser [53] to generate dependency relations for requirements text. The relative position of each token in relation to its head token within the previously established dependency is used as an input feature to extract functional requirements.

k: SENTIMENT ANALYSIS

library in NLTK provides methods for analyzing the sentiment of text, such as determining whether a piece of text is positive, negative, or neutral.

In the study conducted by Kengphanphanit et al. [54] uses Text-blob library from NLTK for sentiment analysis. The library extracts polarity, subjective factors of software reviews. Text-blob library is used to assign values for these factors. Polarity factor indicates the sentiment of user feedbacks which has values in the range of $[-1,1]$. Polarity value that is greater than 0 indicates the mood is ''positive'', polarity value of 0 indicates the mood is ''neutral'', and polarity value that is less than 0 indicates the mood is ''negative''. Subjective factor indicates opinion which has values in the range of [0,1] where the value that is equal or less than 0.50 indicates the sentence is ''objective'' and the value greater than 0.50 indicates the sentence is ''subjective''. These features are used to extract requirements from app reviews.

Dependency Parser Annotator, such as the one found in Stanford CoreNLP, is an integral component of NLP pipeline. It entails the analysis of word relationships within sentences, representing these connections as directed edges in a dependency parse tree structure. Each word serves as a node in this tree, with the directed edges depicting syntactic relationships like subject-verb, object-verb, and modifier-head.

Stanford CoreNLP utilizes linguistic rules to conduct dependency parsing on input sentences. Researchers in Dragoni et al. [21], Villamizar et al. [55], Yang and Liang [56] leverage this library to examine the grammatical structure of sentences, discern word relationships, and generate a dependency parse tree for each sentence.

Coref Annotator, like Stanford CoreNLP, is a component responsible for resolving coreferences in text. Coreference resolution is the task of identifying all expressions in a text that refer to the same entity and linking them together. For example, consider the sentence: ''As a user, I want to be able to access my emails from any device so that I can stay connected and respond to important messages promptly.'' In this sentence, the coreference occurs in the phrase ''my emails''. The pronoun ''my'' refers back to the user, establishing a relationship between the user and the emails they possess. Coreference resolution aims to identify this relationship and link both mentions to the same entity.

The Coref Annotator analyzes text to identify and resolve such coreferences, creating clusters of mentions that refer to the same entity. It employs linguistic and contextual cues to determine which mentions are coreferent and to which entity they refer.

2) TEXT VECTORIZATION TECHNIQUES

Text vectorization is the process of converting preprocessed text into a vector space model that can be used as input for machine learning algorithms. In NLP, text vectorization is a crucial preprocessing step that enables algorithms to process and analyze textual data effectively. In this subsection, various techniques for text vectorization utilized in AI-driven solutions aimed at tackling elicitation tasks, as shown in Figure 4(C), are introduced.

a: BAG-OF-WORDS (BOW)

In this approach, each document is represented as a vector where each dimension corresponds to a unique word in the vocabulary. The value of each dimension represents the frequency of that word in the document. However, BoW ignores the order of words in the document and only considers their frequencies. In a research by Quba et al. [57] initially normalize software requirements from the PROMISE dataset [58]. Subsequently, they utilize the Bag-of-Words (BOW) technique to extract features from these normalized textual requirements, facilitating the classification of software requirements into FR and NFR. Likewise, BOW is also utilized in studies such as, Maalej et al. [59], and Dhinakaran et al. [60].

b: TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY

Different from BOW, Term Frequency - Inverse Document Frequency (TF-IDF) determine the mathematical significance of words in documents. In this approach, represents each document as a vector where each dimension corresponds to a unique word in the vocabulary, and the value of each dimension is the TF-IDF score of that word in the document. TF-IDF weighs the importance of words based on their frequency in the document and their rarity across the entire corpus. The TF-IDF value is defined using the Formula 1.

$$\text{TF-IDF}(word_{a,b}) = f_{a,b} * log \frac{corpus\_size}{total\_documents\_with\_word\_a} \quad (1)$$

where, $f_{a,b}$ denoted the frequency of word a in the document b. obtained by multiplying two values: TF (Term Frequency) and IDF (Inverse Document Frequency). Asif et al. [61] employed TF-IDF as a weighting scheme to convert preprocessed tweet text into a vector space model. Similarly, TF-IDF has been used in studies [11], [16], [36], [47], [48], [49], [56], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71].

c: CHI SQUARED

Similar to TF-IDF, a common statistical test used to select textual features is Chi Squared. Chi Squared approach for feature selection has been used in classification of Functional Requirements and sub-categories of Non-Functional Requirements in Lu and Liang [72]. The Chi Squared is computed using the following Formula 2:

$$\text{CHI}^2(t_a, C_k) = \frac{N * (eh - fg)^2}{(e+g) * (f+h) * (e+f) * (g+h)} \quad (2)$$

The variables e, f, g, and h of Formula 2 are explained in table below, and N refers to number of documents in the dataset.

TABLE 2. Explanation of variables used in Chi-Squared Formula.

Category		Belong to	Do not belong to
Number of documents	With term t_a	e	f
	Without term t_a	g	h

d: NEURAL NETWORK-BASED WORD EMBEDDINGS

Neural network-based embeddings are the dense vector representations of words or entities that map a single word to a vector $v \in \mathbb{R}^n$, where n denotes *embedding size*, that are learnt using neural network architectures. These embeddings capture semantic relationships between words or entities based on their co-occurrence patterns in large text corpora. A remarkable property of word embeddings is that if two words are used in similar context then the vector distance of the two words is small, whereas the vector distance of two unrelated words is large. Neural network-based embeddings are trained in an unsupervised manner, where the model learns to map words or entities to continuous vector representations in a high-dimensional vector space.

A large number of articles have used word embedding for representing requirements documents as fixed-length vectors. There are several popular neural network architectures used to generate word embeddings. FastText, Word2vec, doc2vec and BERT are the most preferred word-embedding models in studies [24], [52], [65], [68], [73], [74], [75], [76], [77]. BERT-based Word embeddings are the base of studies [13], [78], [79], [80], [81], [82], [83]. Similarly, GloVe embeddings is used in Kaur et al. [84] and Skip-gram embeddings is used in Navarro-Almanza et al. [83] to generate word embedding vectors of 400 dimensions for words in the requirements. Additionally, study in Kumar et al. [76] compared the performance of five word embedding techniques, namely Continuous Word2Vec, Skip-Gram (SKG), Bag of Words (CBOW), Fast text and Global Vectors for Word Representation (GloVe) to categorize Functional and Non-function Requirements.

In a study by Kurtanović and Maalej [85], authors use a combination of N-grams and statistical textual feature

engineering techniques to improve the accuracy of their AI model that separates Functional and Non-functional Requirements. Their methodology integrates word embeddings with statistical attributes, including percentage of nouns, verbs, adjectives, adverbs, length of the requirement sentence(s). Similarly, N-grams have been used in Tizard et al. [86], Kurtanović et al. [40], and Devine et al. [87] for text preporcessing.

3) MACHINE LEARNING ALGORITHMS

This section outlines the machine learning algorithms that are predominantly employed in cutting-edge techniques. The algorithms featured in the studies are classified into two categories: supervised learning algorithms and unsupervised learning algorithms. The supervised category encompasses both conventional machine learning algorithms and deep learning methods. In the realm of unsupervised learning, the techniques include methods such as Latent Dirichlet Allocation (LDA), Expectation Maximization (EM), and hierarchical clustering.

a: SUPERVISED LEARNING ALGORITHMS

Among the studies analyzed, various machine learning algorithms have been explored by researchers. For instance, Logistic Regression was employed by Binkhonain et al. [88] in their work on hierarchical classification, while Naive Bayes was utilized by Kadebu et al. [89]. Furthermore, Decision Trees were examined by Hayes et al. [90] However, SVM emerged as the most prevalent classifier among nearly 30 studies, illustrating its popularity among researchers, followed by Tree-based algorithms such as Decision Trees and Random Forest and Naive Bayes approaches that are found in 15 and 17 approaches, respectively. Notably, all of these studies leveraged machine learning classifiers provided by Scikit-learn for implementing their research.

Furthermore, while Scikit-learn served as the primary tool for machine learning classifiers, some researchers also made sparing use of ML Classifier available within Weka API tool. Weka, a widely-used suite of machine learning software written in Java, provided an alternative avenue for experimentation, albeit less frequently explored in comparison to Scikit-learn.

In the realm of neural networks, Convolutional Neural networks have been used in approximately 30 studies. Moreover, a range of different architectures has been examined in the studies included in this review. Shallow neural networks and Bi-LSTM models have been prominent choices among researchers, demonstrating their versatility and effectiveness in tackling elicitation tasks. However, BERT, a state-of-the-art language representation model, has garnered significant attention and popularity among researchers in recent years.

Among the studies examined, BERT emerges as the best classifier for addressing elicitation tasks. It has consistently demonstrated impressive performance, achieving an average F1-score of approximately 0.91 across all tasks it has been applied to. This remarkable level of accuracy underscores BERT's efficacy and robustness in handling a diverse range of natural language processing challenges, further solidifying its position as a leading choice for researchers in the field.

b: UNSUPERVISED LEARNING ALGORITHMS

The included studies also contain unsupervised algorithms: cluster-based solutions and topic models. Clustering refers to the task of organizing items into groups such that objects in a cluster are more similar to one another than the objects in other clusters. Hierarchical clustering which is a type of Clustering algorithm is utilized in four studies.

Approaches in Bhowmik et al. [91], Jiang et al. [92] Jeong et al. [38] are based on Latent Dirichlet Allocation (LDA) on the comments from stakeholders. Comments posted by a group of stakeholders is collected as text documents. After which, LDA is applied on each document to produce topic-word distribution matrix and document-topic distribution matrix which are used to extract creative requirements. In Arora et al. [51] and Bhatia et al. [52] authors experimented with three popular clustering algorithms, K-Means, Agglomerative and Expectation Maximization (EM) to cluster candidate glossary terms based on the degree of relatedness. The input to the clustering algorithms is a similarity matrix that captures the degree of similarity between every pair of candidate glossary terms, and K, which indicates the number of groups to generate. The LDA and Agglomerative clustering algorithms are also used in Abad et al. [93] for classifying sub-categories of Non-functional Requirements, and in Li et al. [94] to group requirements that have a same theme, also called as 'topic' to extract requirements.

Shambour et al. [95] uses hybrid content-based collaborative filtering recommendation approach to search project-related requirements from different sources and recommend related requirements to project stakeholders while mitigating the risk of missing requirements. Mezghani et al. [30] employs k-means algorithm for identifying redundant and inconsistent Requirements. Their algorithm partitions a given set of requirements into a predefined number of k clusters. A study by Dopaz et al. [96] uses Decision Trees to predict the quality of requirements. If a requirement is predicted as poor, then their approach proposes a technique to enhance the requirement using Genetic algorithms.

Sonbol et al. [31] proposed an unsupervised approach to identify templates that are used to standardize the structure of requirements such as Rupps, EARS, and User Stories. Their approach uses graphs to represent requirements and uses dynamic programming to detect communities in the graph which are later interpreted as templates. Akay and Kim [97] employs BERT, fine-tuned on SQuAD by the HuggingFace Transformer library to build a map of the functional domain of the problem by extracting high-level functional

requirements(FR) and decompose them into low-level Design Parameter (DP), which represent ''how'' the design achieves ''what'' the FR defines.

Chen et al. [98] employed online learning to create an intelligent assistant system 'CitySpec'- a translation system converting human-specified requirements to machine-understandable formal requirement specification in smart cities. The authors extracted city-related knowledge using 1,500 real-world city requirements and built a database of city specific vocabulary with 3,061 words. Their also explored the role of online learning to enhance requirements through requirement synthesis. Their aim is to support policy makers to create and refine requirements.

Case Based Reasoning CBR is a branch of Artificial Intelligence that has attracted significant interest from a wide range of research domains and it has been successfully used in Software Engineering for prediction tasks. CBR is used in Sarro et al. [99]. CBR identifies similar apps by using a similarity function that measures the distance between the target case and the other cases based on the values of the n features of these apps.

The original BERT model randomly selects 15% of the words in the training text for a masking operation, which is a character-level masking method. But authors in Li et al. [100] introduced 'N-gram Masking-based BERT Word Embeddings Model' into BERT and use N-gram to modify the original BERT masking method so that different combinations of characters are considered during training. Specifically, we designed a variable-length N-character sliding window that can mask out N characters at once for each masking operation. These masked text is used for NFR Classification.

B. RQ2: DATA SOURCES FOR MODEL TRAINING

This section presents several widely-used publicly accessible datasets suitable for training a machine learning models for RE tasks. The twelve publicly available data sources are highlighted in Fig. 4(A).

Most studies obtain their data from software reviews on social media platforms such as Facebook and WeChat(e.g. [101]). Data preparation involves crawling websites to gather relevant information. Additionally, reviews of Open Source Software (OSS systems) are also used to source data. For instance, Morales-Ramirez et al. [102], and Bhowmik et al. [91] acquired a dump of user feedback for Apache OpenOffice (AOO) and Bugzilla to compile their training data. Guzman and Emitza [62] collected tweets related to well-known software applications- Spotify, Dropbox and Slack from Twitter platform to identify requirements. Reddit posts for the software applications are Netflix, Google Maps, Amazon, Excel, Spotify, and Whatsapp are used as data sources for mining requirements in Iqbal et al. [11] and Khan et al. [12]. Reviews of applications hosted on Google Play Store and Apple App Store have also been used for extracting requirements from software reviews. Commonly analyzed applications include Netflix, WhatsApp, Twitter and Spotify. Moreover, descriptions of applications from from Google Play have also been employed to build feature recommendation systems. Authors also utilize tools such as Google Play Unofficial and the open-source API 'AppReviews' to efficiently construct their dataset. Software distribution platforms (SDPs), such as Softpedia, sourceforge.net and toptenreviews.com, feature software product listings. These platforms not only offer details regarding download and installation options but also highlight key features that may appeal to potential users. Consequently, SDPs serve as valuable resources for authors in [34] and [36] for gathering requirements. Additionally, the dataset is acquired from meta.stackoverflow.com, a platform for user feedback on Stack Overflow [103]. User manuals of scientific software from three different scientific domains, namely seismology, building performance and computational fluid dynamics have also been in [94] to automatically extract scientific requirements. The intention is to easily recover and reuse requirements without acquiring prior requirements engineering knowledge. Common Weakness Enumeration (CWE) dataset [104] is a community-developed. It does not contain requirements, but is a list that describes software and hardware weaknesses. Extended CWE (ExCWE), as the name suggests, contains extended description of software vulnerabilities that has been used in identifying security requirements by [14].

Legal documents such as General Data Directive (EU) 2019/770, Directive (EU) 2019/771, Protection Regulation (EU) 2016/679, Luxembourg Law of 25 March 2020 are used for extracting compliance requirements. Recordings of meetings between developers and customers are used as data source (e.g. Rodeghero et al. [105]). In Agile software development, customers articulate their needs to the technical team members who take notes during conversations. These notes are reread to ask clarifying questions and highlight crucial problems in their requirements. The transcripts of conversations are annotated in the work, Rodeghero et al. [105] for automatically extracting user stories.

Typically, gathering data poses no significant challenge for researchers. However, there is a high probability that the amassed data is unlabelled, which, for supervised machine learning tasks, is nearly worthless. The main constraint lies in the necessity for authors to manually annotate the dataset for training, a process that is both laborious and time-consuming. CrowdRE contains the requirements that are gathered through crowd-based elicitation techniques via interviews and similar techniques. The generated ideas in this repository have been used for glossary extraction.

An alternative is to utilize crowd-sourcing for the labeling task, a strategy employed by [106]. The authors chose to Amazon Mechanical Turk (AMT) emerges as one of the largest crowd-sourcing platforms, offering access to an extensive pool of crowd-workers via the Internet to carry out human-intelligence tasks. Researchers engage

crowd-workers to annotate the training data or to share their interaction data with applications they commonly use, to generate data.

Annotation software-Snorkel [107] is a tool that we can use to automate labeling data. Snorkel is a data labeling and machine learning framework designed to assist in creating training datasets for machine learning models, especially in scenarios where obtaining labeled data is challenging or expensive. Snorkel allows users to define labeling functions, which are functions that heuristically label data. These functions can be simple rules, patterns, or even weak models. Labeling functions don't need to be perfect but should provide some signal about the labels of the data. After defining labeling functions, they can be applied on the unlabeled dataset using Snorkel's labeling interface, after which Snorkel will then automatically label your data based on the outputs of these functions. Thus by using Snorkel, researchers can leverage weak supervision to generate training data at scale, even when obtaining manually labeled data is impractical or expensive. This helpful library is used in [108] to label data for NFR Classification.

Publicly available labeled data do indeed exist, including well-known ones like NFR-PROMISE, PURE (PUblic REquirements), and PROMISE_exp. These datasets are invaluable resources as they come pre-labeled, which greatly facilitates various elicitation tasks. For instance, they are ideal for extracting glossary terms, identifying features, and categorizing both functional and non-functional requirements. Moreover, they offer the advantage of being suitable for classifying subcategories within non-functional requirements. Each of these datasets typically contains numerous documents outlining requirements, conveniently formatted in plain text for ease of use and analysis. Moreover, SemEval orchestrates workshops aimed at presenting and evaluating a diverse range of shared tasks, each highlighting semantic analysis systems crafted by different teams. These tasks span a broad spectrum, including semantic role labeling, semantic relation analysis, and co-reference resolution. These workshops provide a platform for researchers to exchange ideas, compare methodologies, and advance the field of semantic analysis. The datasets from SemEval-2015 and 2016 are derived from reviews of software applications, that served as corpus for researchers involved in aspect extraction task. Additionally, DePaul University has published PROMISE corpus- a compilation of software requirement specifications for term projects completed by MS students, covering a wide range of topics. These specifications encompass a total of 326 non-functional requirements and 358 functional requirements. The subcategories of NFRs include availability, look-and-feel, legal considerations, maintainability, operational requirements, performance benchmarks, scalability, security measures, and usability criteria. Notably, the dataset comes pre-labeled, facilitating its immediate use in classification tasks such as distinguishing between functional and non-functional requirements and identifying different types of non-functional requirements. This dataset serves as a valuable resource for researchers and practitioners alike in the field of software engineering and requirement analysis.

In many studies, the number of instances belonging to one class (the minority class) was significantly smaller than those belonging to another class (the majority class). This class imbalance leads to poor performance of machine learning algorithms, particularly those that are sensitive to class distribution. To overcome this issue, oversampling technique Synthetic Minority was used, for instance in [11] and [12]. It is a technique used in the field of machine learning, particularly in dealing with imbalanced datasets by generating synthetic examples for the minority class. It works by creating new instances that are similar to existing minority class instances. This is done by randomly selecting a minority class instance and finding its nearest neighbors. New instances are then created by interpolating between the selected instance and its neighbors.By synthesizing new instances, SMOTE helps balance the class distribution in the dataset, making it more suitable for training machine learning models. This generally leads to better performance, especially for classifiers that struggle with imbalanced data.

C. RQ3: WHAT ARE THE PEAK PERFORMANCE METRICS ACHIEVED BY THE AI TECHNIQUES?

This subsection presents the highest performance achieved by AI-based solutions in solving various elicitation tasks. A complete list of AI-based approaches along with their precision, recall, F1-score, and accuracy is highlighted in Table 4. In the task of classifying requirements into Functional Requirements and Non-Functional Requirements, Decision Trees in [90] and Naive Bayes in [89] achieved accuracies of 0.9 and 0.86, respectively. Notably, BERT, a deep-learning algorithm considered to be more sophisticated was used in [109]; it achieved an accuracy of 0.98. For mining requirements from application reviews, Random Forests used in [62] reached the highest accuracy of 0.74. In the classification of various categories of Non-Functional Requirements, Support Vector Machines in [85] achieved an accuracy of 0.72. When it comes to extracting intentional elements from requirements, a neural network-based solution in [33] achieved an accuracy of 0.47. Additionally, the highest accuracy achieved for User Story extraction from natural language requirements was 0.58 in [15]. For Feature Extraction, an NLP-based technique in [22] achieved an accuracy of 0.55. In the task of recommending missing requirements, an AI-based approach in [95] achieved a Mean Absolute Error (MAE) of 29.3. Finally, for extracting glossary terms, the approach in [51] achieved an Area Under the Curve (AUC) score of 0.547. Overall, these results demonstrate the varying levels of success different AI algorithms have achieved across a range of requirements elicitation tasks.

D. RQ4: WHAT ARE THE STRENGTHS AND LIMITATIONS OF THE CURRENT AI METHODS?

This subsection presents the strengths and weaknesses of the AI methods that are currently employed in addressing challenges in requirements elicitation tasks. As detailed in Table 4, popular machine learning algorithms in the surveyed studies are SVM, Decision Trees, and Logistic Regression. These algorithms are favored for their interpretability, ease of implementation, and efficiency in handling structured data [110].

SVM is known for its robustness in high-dimensional spaces and its effectiveness even when the number of dimensions is larger than the number of specimens [110]. One of the significant strengths of SVM is its capability to work well with unstructured and semi-structured data, such as text, images, and trees. The kernel trick is a real strength of SVM, allowing it to solve complex problems by modeling non-linear decision boundaries. By using an appropriate kernel function, SVM can map the data into a higher-dimensional space where the data becomes linearly separable [110]. Additionally, SVMs are robust to noise, as the decision boundary is determined by the support vectors, which are the closest data points to the boundary. Decision Trees are also well-adopted by the research community because they provide clear decision paths that are easy to understand and visualize, making them suitable for interpretability. Logistic Regression, although simpler, remains effective for binary classification tasks and provides probabilistic interpretations of the outcomes.

Although traditional machine learning techniques are widely used, they have notable limitations. For instance, training SVMs can be computationally intensive, particularly with large datasets, due to the quadratic programming required to find the optimal hyperplane [111]. Additionally, SVMs require significant memory to store support vectors and their performance heavily depends on the choice of kernel and its parameters, which can be challenging to tune [111]. SVMs are also sensitive to outliers and noise, affecting hyperplane placement and reducing model performance. Similarly, decision trees are prone to overfitting, capturing noise in the training data. The recursive splitting algorithm they use is greedy, which may not always lead to the globally optimal tree structure [111].

On the other hand, deep learning algorithms based on Artificial Neural Networks (ANNs) have also been widely used in the studies. ANNs are particularly powerful in handling unstructured data and capturing complex patterns that traditional machine learning algorithms might miss. One of the biggest advantages of deep learning is its capacity to identify non-linear relationships in data, a task that previous methods find difficult and complex to discern. Their ability to automatically extract features from raw data reduces the need for manual feature engineering, making them suitable for more complex requirements elicitation tasks. However, deep learning models often require large datasets and substantial computational resources for training, which can be a limitation in some contexts [112].

Several studies highlighted in Table 3 attribute their subpar results to the poor quality of existing data. For instance, [113] cite errors in labeling and the presence of duplicated data as significant factors negatively impacting their outcomes. Similarly, [32] state that using predefined security requirements keywords and linguistic-rule based approaches limits the generalizability of their solution. Additionally, [74] note that their model is only effective for their dataset due to specific data processing techniques they used. The approach in [64] is also limited, as it fails when search text spans multiple regulatory articles within the same document. These issues introduce noise and inconsistencies into the training process, leading to less accurate and reliable models. Collectively, these challenges underscore the critical importance of high-quality data for the success of machine learning models.

Before applying any machine learning model, the studies vectorized the text first using NLP techniques. This preprocessing step transforms raw text data into numerical representations that machine learning models can process. The common vectorization pipeline included following steps: tokenization, stemming or lemmatization, removing stop words, and using embedding techniques such as GloVe or BERT. These heuristic rules lack contextual understanding of natural language requirements, causing them to overlook nuances and leading to the misinterpretation of user requirements.

This study also conducted a survey of different limitations of the state-of-art solutions which has been summarized in Table 3. The study categorized the limitations of state-of-art techniques into following classes - 'low generalizability', 'suboptimal results despite sufficient data', 'lack of data to get good results', 'needs manually annotated data', 'using unbalanced dataset', 'poor data quality', 'needs high computational resources', and, 'suboptimal results'. This study finds that 'limited generalizability' is the topmost drawback of the state-of the art techniques, followed by 'suboptimal results despite sufficient data'.

V. FUTURE RESEARCH DIRECTIONS

Automating the requirement elicitation phase using AI remains a significant challenge. Firstly, requirement elicitation encompasses a multitude of tasks, as outlined by Wiegers and Beatty [114], which are completed iteratively and collectively constitute the elicitation phase. For example, these tasks range from specifying scope to ensuring requirements reusability, making it arduous to construct a pipeline of ML-based solutions capable of addressing the diverse array of tasks involved. Secondly, unlike rule-based systems that can be easily debugged, ML-based approaches pose challenges in debugging due to their automatic feature learning capabilities. This inherent complexity makes it difficult to analyze and diagnose errors effectively. Additionally,

TABLE 3. Summary of drawbacks of state-of-art techniques.

Drawback	References
Needs high computational resources.	[59], [61], [82], [117]
Suboptimal results	[12], [15], [18], [21], [22], [24], [33], [36]–[38], [43], [44], [48], [49], [52], [54], [57], [60]–[63], [65], [67], [70], [82], [89], [105], [108], [118]–[126]
Lack of data to get good results	[11], [13], [17], [28], [31], [67], [74], [81], [83], [90], [102]
Suboptimal results despite sufficient data.	[12]–[16], [18], [21], [24], [33], [37], [38], [44], [47], [49], [52], [54], [62], [66], [70], [76], [78], [80], [86], [88], [92], [96], [102], [108], [113], [118], [120], [127]–[129]
Needs manually annotated data	[11], [12], [27], [48], [52], [57], [63], [65], [70], [72], [81], [86], [91], [103], [113], [117], [123], [126], [129]–[138]
Using unbalanced dataset	[16], [24], [47], [54], [60], [72], [89], [93]
Poor data quality	[60], [74], [88], [93]
Low generalizability	[11], [15]–[17], [19]–[21], [23]–[33], [35]–[38], [41], [43]–[46], [48], [49], [51], [52], [54], [55], [57], [63]–[66], [68], [69], [71]–[81], [84]–[86], [88], [90], [91], [94], [95], [100], [103], [106], [109], [113], [117]–[119], [121], [122], [127], [132], [134], [139]–[152]

the current research landscape predominantly focuses on classification and clustering techniques, resulting in a lack of diversity in approaches. To address these challenges, several future research directions are proposed.

First, the recent advancements in Large-Language Models (LLMs) can be revolutionary in addressing several RE-related challenges. LLMs, such as GPT and Llama, are trained on vast amounts of text data and can capture complex patterns and contextual relationships within language. They have the ability to understand and also generate text based on the context provided, enabling more accurate and contextually relevant language processing. Specification tasks such as, transforming Natural Language requirements into structured templates such as, EARS can be achieved using LLMs.

Second, in agile software development, user stories are the most commonly used method to elicit users' needs. These stories are in natural language format and therefore it is difficult to establish relationships between requirements. In this context, goal modeling is vital for interpreting relations among goals and requirements. LLMs can be employed to automatically generate goal models from user stories. Similarly, LLMs can be employed to extract Activity Diagrams, and Class Diagrams from natural language requirements or semi-structured user stories.

Third, AI-based methodologies rely heavily on data, necessitating high-quality requirements for optimal performance. While there are public requirements datasets available, such as PURE, PROMISE, and PROMISE NFR, these datasets are relatively small to train deep neural networks. One approach to address this challenge is to share anonymized industrial requirements with the research community. Additionally, leveraging Large Language Models presents another viable strategy for synthetically generating requirements, thereby circumventing the need to disclose sensitive information while still providing valuable data for research and development in requirement engineering. These approaches offer promising solutions to the dilemma of balancing data privacy with the advancement of AI-driven methodologies in the field of requirement engineering.

Fourth, there are large volumes of textual requirements in the public domain, but they are often unannotated, making it difficult to extract useful insights or patterns from them efficiently. LLMs can help facilitate the annotation of textual requirements, making the requirements engineering process more efficient and scalable. They can classify textual requirements into predefined categories, such as functional vs. non-functional requirements and can also detect the underlying intent behind a requirement, such as whether it's a request for a feature, a performance expectation, or a security constraint. This helps in prioritizing or grouping requirements during analysis.

Fifth, there is a pressing need to develop language models specifically tailored for tasks within requirements engineering. Studies indicate the applicability of Transfer Learning techniques in adapting models initially trained in one domain, such as, image classification, to related domains, like melanoma image classification [115]. This approach not only mitigates training time but also proves valuable in scenarios with limited data availability in the new domain. Consequently, neural language models tailored for Requirements Engineering tasks can be effectively repurposed to address various tasks within the domain. This versatility underscores the potential of Transfer Learning in enhancing the efficiency and effectiveness of machine learning approaches across diverse domains and applications within RE.

Sixth, incorporating explainability into AI-generated predictions is essential for building trust, and enabling stakeholders to verify the decisions made by AI systems, particularly in complex domains such as, requirements engineering. Visual tools that show the decision path the model took to reach a prediction can be helpful. For example, a decision tree

TABLE 4. Summary of all state-of-art techniques on the basis of three parameters, Accuracy (A), Precision (P), Recall (R), and F1-Score.

Reference	Elicitation Task	Techniques used	Dataset	Publicly available	Accuracy	Precision	Recall	F1-score
[82]	Mining requirements	Lemmatization, stopword removal, remove special characters, Multilingual DistilBERT	Dataset prepared by Maalej et al. [59]	✓	-	-	-	0.52
[82]	Mining requirements	Tokenization, IR Techniques-Heuristic	SemEval 2015	✓	-	0.69	0.54	0.60
[61]	Requirements Extraction	Tokenization, stemming, stopword removal, remove special characters, lowercasing, Multinomial Naive Bayes	SRS from DePaul University	✓	0.89	0.22	-	-
[13]	Classify FR and NFR	NoBERT	PROMISE NFR	✓	-	-	0.87	-
[13]	Classify FR and NFR	Deep Neural Network	PROMISE	✓	-	0.81	0.79	0.77
[105]	Detect user story information	Support Vector Machines and Logistic Regression	AMI meeting corpus	✓	-	0.76	0.25	-
[102]	Mining requirements	Remove special characters, Random Forests, J48 & SMO	Discussions from Open Office Software		-	0.7	0.66	0.68
[62]	Mining requirements	Tokenization, stemming, stopword removal, lowercasing, Multinomial Naive Bayes, Random Forest	Tweets		-	0.74	0.79	0.79
[85]	Classify FR and NFR	Lemmatization, stopword removal, remove special characters, Support Vector Machines	NFR-dataset	✓	-	0.72	0.90	-
[59]	Classify FR and NFR	Naive Bayes	Reviews from Apple App Store and Google Play Store		-	0.86	0.83	0.85
[72]	Classify subcategories NFR	Stemming, stopword removal, Naive Bayes, J48 decision tree, Bagging	Reviews from Apple App Store and Google Play Store		-	0.71	0.72	0.72
[93]	Classify FR and NFR	Part Of Speech tagging, J48 decision tree	PROMISE [153]	✓	-	0.95	0.94	0.94
[113]	Classify subcategories NFR	Naive Bayes, Decision Tree, Logistic Regression, J48 decision tree	SRS from DePaul University	✓	-	0.7	0.7	0.76
[14]	Classify subcategories NFR	TF-IDF, Support Vector Machine	CWE [104]	✓	-	0.67	0.70	0.67
[74]	Classify FR and NFR	Lemmatization, stopword removal, lowercasing, Convolutional Neural Networks	Private		-	0.73	0.89	-
[117]	Mining requirements	Stemming, stopword removal, Naïve Bayes, J48 decision tree, Support Vector Machine, Nearest Neighbor, and AdaBoost	Private		-	0.69	0.81	-
[90]	Classify FR or NFR	Decision Trees	Bay Area 511 Regional Real-Time Transit Information System [154]		0.9	-	-	-

TABLE 4. *(Continued.)* **Summary of all state-of-art techniques on the basis of three parameters, Accuracy (A), Precision (P), Recall (R), and F1-Score.**

[64]	Mining requirements	Tokenization, lemmatization, stopword removal, remove special characters, BERT	General Data Protection Regulation		0.94	-	-	-
[27]	Mining NFRs	Tokenization, POS Tagger, Random Forest	30 industrial requirements specifications		-	0.8	0.9	-
[142]	Classify FR and NFR	Tokenization, stemming, stopword removal, Logistic Regression	SRS from DePaul University		-	0.1	0.7	-
[28]	Requirements Classification	Tokenization, Lemmatization, stopword removal, POS Tagger, SVM	PROMISE	✓	-	0.87	0.89	0.98
[29]	Classify requirements into topics	Tokenization, POS Tagging, Naive Bayes	Real software development fields		-	0.87	0.82	0.84
[113]	Classify NFR	J48	Three industrial requirements documents	✓	-	-	-	0.77
[88]	Multiclass requirement Classification	Tokenization, lemmatization, stopword removal, LogisticRegression & Decision Trees	PROMISE-exp	✓	-	0.61	0.48	0.51
[146]	Classify FR anf NFR	Sentence-BERT (Sbert)	PROMISE NFR & SecReq dataset	✓	-	0.71	0.66	0.66
[32]	Detecting privacy requirements from User Stories	Tokenization, POS Tagger, shallow neural networks	1680 user stories	✓	0.72	-	-	0.71
[33]	Extract actions and actors from requirement	Tokenization, POS Tagger, neural network	5 use cases		-	0.47	0.79	0.55
[145]	Classify NFR categories	Lemmatization, stopword removal, remove special characters, lowercasing, BERT	SoftwareNFR	✓	-	0.91	0.92	0.91
[144]	Relation extraction	Tokenization, remove special characters, lowercasing, CNN-based Model	Occupational Safety and Health Administration [155]	✓	-	0.87	0.83	0.85
[98]	Translate requirements into specifications	BERT	1,500 real-world requirements		-	0.81	0.78	0.84
[120]	Multilabel Requirements Classification	Tokenization, remove special characters, lowercasing, Bi-GRU	PROMISE & EHR	✓	0.57	0.49	0.42	0.44
[109]	Classify FR and NFR	BERT with CNN	PROMISE [58] & EXP [156]	✓	-	0.98	0.94	0.96
[89]	Classify FR and NFR	Multinomial Naive Bayes	DOSSPRE [157]	✓	0.86	0.91	0.69	0.78
[65]	Mining reviews for requirements	Lemmatization, stopword removal, SVM	Online reviews		0.77	0.92	0.81	
[16]	Relation extraction	Lemmatization, stopword removal, lowercasing, SVM	PROMISE_exp [156]	✓		0.71	0.72	0.71
[36]	Feature extraction	Tokenization, POS Tagger, K-Means Clustering	Online reviews	✓	-	0.62	0.82	0.70
[75]	Classify FR and NFR	Tokenization, lemmatization, stopword removal, CNN model	Standard specification of the US Department of Transport	✓	-	0.93	0.91	0.92
[140]	Quality requirement classification	CNN model	Aurora [158]	✓	0.87	-	-	0.90
[39]	SBVR to CRC	Tokenization, POS Tagger, Rule based	Data used in [159]	✓	-	0.94	0.96	-
[141]	Feature extraction	Rule based	Data from [160]	✓	-	-	-	0.76
[77]	Ambiguity detection in requirements	Tokenization, expand contractions, CNN	Privacy policies of Android apps [161]	✓	-	-	-	0.90
[143]	Mining Requirements	BERT-baseuncased	Software feedbacks from social media channels [138]	✓	-	0.91	0.91	-

TABLE 4. *(Continued.)* **Summary of all state-of-art techniques on the basis of three parameters, Accuracy (A), Precision (P), Recall (R), and F1-Score.**

[11]	Mining requirements	Tokenization, lemmatization, stopword removal, remove special characters, SVM	Reddit posts for software applications		-	-	-	0.84
[130]	Feature extraction	SBERT	Feedback channel of a telecommunication company on Twitter		Unsupervised			
[86]	Mining Requirements	MLClassifier from Weka API [162]	Feedbacks for VLC media player and Firefox web browser forums	-	-	-	0.78	
[12]	Mining requirements	Tokenization, lowercasing, SVM	User forum data from Reddit		-	0.74	0.95	0.83
[55]	Classify NFR categories	Rule based technique	Undergraduate computer science projects [163]		Unsupervised			
[108]	Classify NFR categories	BERT	PROMISE [58], SRS documents available at [164], and private data	✓	-	-	-	0.56
[100]	Classify NFR categories	Bi-LSTM	PROMISE [58]	✓	-	0.91	0.92	0.91
[147]	Classify FR and NFR	Bayesian classifier	Private dataset	✓	0.67	0.55	0.73	-
[127]	Identify uncertainty in requirements	Conditional Random Fields	11 requirements documents from RE@UTS web pages [165]	✓	-	0.61	0.62	0.61
[67]	Classify feature models	Tokenization, LIBSVM	Private dataset		-	0.92	1	-
[118]	Mining requirements	Tokenization, stemming, stopword removal, remove special characters, SVM	4,000 tweets sampled from Twitter feeds of software systems		-	0.79	0.75	0.77
[40]	Mining requirements	Lemmation, stopword removal, POS Tagging, Naive Bayes	reviews for 52 software applications in the Amazon Store	-	0.83	0.98	0.81	
[68]	Classify FR and NFR	Lemmatization, stopword removal, lowercasing, MultinomialNB	SecReq [166] and NFR dataset [145]	✓	0.86	0.86	0.86	0.86
[85]	Classify NFR and FR	Tokenization, lemmatization, stopword removal, Support Vector Machine	Dataset provided by RE17 data challenge	✓	-	0.92	0.93	0.93
[85]	Classify NFR subcategories	Support Vector Machine	Dataset provided by RE17 data challenge	✓	-	0.93	0.92	0.92
[119]	Mining Requirements	Lemmatization, stopword removal, Naive Bayes	App reviews from Apple AppStore and Google Play		-	0.90	0.8	-
[99]	Feature extraction	Case Based Reasoning (CBR)	7 apps from the Samsung Android and BlackBerry World app stores [167]	✓	1	-	-	-
[84]	Classify NFR subcategories	Tokenization, stemming, stopword removal, self-attention based bidirectional LSTM	34 industrial requirements specifications, PROMISE	✓	0.96	0.95	0.96	0.96
[69]	Mining requirements	Lemmatization, stopword removal, Rule-based techniques	Android apps		-	0.97	-	-
[60]	Mining requirements	Lemmatization, stopword removal, Logistic Regression	Data created by [168]	✓	-	0.66	0.65	0.65
[42]	Detect redundancy and inconsistency in requirements	Lemmatization, stopword removal, remove special characters, POS Tagger, k-means			Unsupervised			
[43]	Automatic Glossary Term Extraction	Tokenization, POS Tagger, Rule-based techniques	Unsupervised		-	0.73	0.74	-

TABLE 4. *(Continued.)* **Summary of all state-of-art techniques on the basis of three parameters, Accuracy (A), Precision (P), Recall (R), and F1-Score.**

[70]	Mining requirements	Lemmatization, stopword removal, remove special characters, SVM	10M tweets about 30 different desktop and mobile software applications	-	0.74	0.76	0.75	-
[128]	Feature extraction	Tokenization, stemming, remove specil characters, MNB	GitHub open-source projects:*c geo,LighttpdRadiant,Redmine*	✓	-	0.05	1	-
[44]	Requirements terminology extraction	Lemmatization, stopword removal, remove special characters, POS Tagger, TextRank	Smart home requirements [169], [170]			Unsupervised		
[129]	Mining requirements	Long Short Term Memory (LSTM)	Ardupilot dataflashlogs from Ardupilot UAVs including Iris 3DRs, America Hexcopter Spreading Wings S900, and Intel Aeros		0.87	0.68	1	0.81
[87]	Clustering user feedbacks	Tokenization, lemmatization, stopword removal, HDBSCAN	App reviews from Google Play Store, the Apple App Store, Twitter, and user product forums [86], [118], [123], [132], [134], [150]	✓		Unsupervised		
[71]	Classify FR and NFR	SVM	Functional and non-functional requirements (FR and NFR) [171]	✓	-	0.91	0.97	0.95
[106]	Feature extraction	Linear SVC	Features 170 unique Android apps		-	0.62	0.83	0.70
[133]	Mining requirements	BERT	Legal contracts of vendor organization		-	0.91	0.94	0.93
[23]	Template-Recommender for Legal Requirements	POS Tagger, Rule-based techniques	Labor and health laws of Luxembourg	✓	-	0.90	0.94	
[45]	Use Case Elements to Textual Specification	Tokenization, POS Tagging, linear classifier with SGD	28 different types of textual specifications [166]	✓	-	0.72	0.51	0.57
[46]	Feature extraction	Lemmatization, stopword removal, POS Tagging, LDA	User reviews from the US App Store and Google Play [172]	✓	-	0.58	0.52	0.54
[151]	Classify NFR subcategories	KNN classifier	Six different documents from healthcare domain [173]–[175], [175]–[177]		-	0.82	0.79	.80
[131]	Keyword Extraction from Requirements	Stemming, stopword removal, TextRank technique and inverse frequency analysis	NFRs, obtained from five free online datasets [153], [178]–[181]	✓		Unsupervised		
[81]	Mining requirements	BERT	8 open-source projects: ActiveMq, Archiva, Aspectj, HDFS, Hibernate ORM, Log4j Mopidy, SWT		0.83	-	-	-
[48]	Mining requirements	Random Forest, TF-IDF, stemming, tokenization, POS tagger	data from [72], [119]	✓	0.88	0.82	0.63	0.71
[95]	Recommend missing requirements	Content-based collaborative filtering	RALIC dataset [182]	✓		$MAE = 29.3$		
[21]	Aspect extraction	Stopwords, lemmatization,SenticNet, stopword removal, WordNet lexical database , Pos Tagger, Rule based techniques	SEMVAl datasets [183], [184]	✓	0.85	0.68	0.53	0.60

TABLE 4. *(Continued.)* **Summary of all state-of-art techniques on the basis of three parameters, Accuracy (A), Precision (P), Recall (R), and F1-Score.**

[121]	Classify NFR subcategories	Tokenization, stemming, spell correction, Case conversion, stopword removal, remove special characters, TF-IDF, SVM	PROMISE [156]	-	0.73	0.73	0.72	
[121]	Classify FR and NFR	Tokenization, spell correction, Case conversion, stopwords, TF-IDF, SVM	PROMISE [156]	-	0.91	0.91	0.91	
[57]	Classify NFR subcategories	BOW, SVM	PROMISE_exp [156]	✓	-	0.91	0.91	0.91
[57]	Classify FR and NFR	BOW, SVM	PROMISE_exp [156]	✓	-	0.68	0.67	0.66
[122]	Classify NFR subcategories	Stopwords, bag-of-words, lowercase, Neural Networks	Requirements Engineering Conference's 2017 Data Challenge dataset [185]	✓	-	0.82 to 0.94	0.76 to 0.97	0.82 to 0.92
[186]	Mining requirements	Roberta	Data from [187]	✓	-	0.78	0.74	0.75
[92]	Feature extraction from App Descriptions	Tokenization, stemming, stopword removal, LDA	Descriptions of apps hosted on Apple App Store, Blackberry App World, and Google Play			Hit15 score upto 78.68%		
[22]	Feature Extraction	POS tagger Penn Treebank, remove special characters,stop word removal, lemmatization	App descriptions of 10 apps from Apple App Store [168]	✓	-	0.55	0.43	0.45
[152]	Mining requirements	Bag-of-Frames,SVM	Internal dataset and data from [119], [125]	✓	-	0.94	0.99	0.96
[51]	Extract Glossary Terms	Text chunking, POS tagger, EM	Private dataset			AUC=0.547		
[56]	Classify FR and NFR	Tokenization, stopword removal, remove special characters, stemming, TFIDF, rule-based classifier	Reviews of iBooks APP in English App Store		-	0.82	0.69	0.75
[17]	User stories extraction	Tokenization, POS Tagger, Rule based techniques	Collection of online news			Unsupervised		
[81]	Identify requirements	Tokenization, remove special characters, lowercasing, BERT	8 open-source projects: ActiveMq, Archiva, Aspectj, HDFS, Hibernate ORM, Log4j Mopidy, SWT		0.85	0.91	0.83	-
[19]	Glossary term extraction	Tokenization, lowercasing, remove special characters, POS Tagging, Text chunking, FastText, cosine similarity	CrowdRE dataset [169]	✓		unsupervised		
[15]	User story extraction from NL requirements	POS tagging, chunking, NER, dependency parsing, WordNet, and BloomSoft, rule-based	PURE dataset [188]	✓	-	0.58	0.57	0.57
[18]	Extract software requirements	Tokenization, lemmatization, stop word removal, POS tagger,	Online News		-	0.56	0.57	-
[126]	Requirements classification	Tokenization, stopword removal, stemming, and lemmatization, TF-IDF, Naïve Bayes	App reviews were collected from iBooks (Apple App Store), and WhatsApp, TripAdvisor from Google Play		-	0.62	0.62	0.62

TABLE 4. *(Continued.)* **Summary of all state-of-art techniques on the basis of three parameters, Accuracy (A), Precision (P), Recall (R), and F1-Score.**

[30]	Redundancy and Inconsistency Detection	Remove stopwords, remove special characters, Part-Of-Speech (POS) tagger, noun chunking, bag-of-words, K-means clustering	22 industrial specifications		-	Number of relevant clusters = 92		
[94]	Extract requirements	Remove stopwords, Lemmatization,bigrams, LDA	Scientific projects from computational fluid dynamics, seismology and building performance		0.97	-	-	-
[54]	Extract requirement	Remove incorrect words, emoticons, and no-meaningful words, Text-blob, Naive Bayes	600 user feedbacks from both Facebook and Twitter		0.65	0.51	0.81	0.63
[135]	Detect Causality in Requirements	Tokenization, BERT	Dataset provided by [189]	✓	0.82	0.84	0.79	0.81
[76]	Requirements classification	Tokenization, lemmatization, stopword removal, remove special characters, SVM	PROMISE [58]	✓	-	-	-	0.73
[136]	Requirements classification	SVM	Industrial requirements	✓	-	0.64	0.80	-
[49]	Software Requirement Recommender	Tokenization, lemmatization, stopword removal, SVM	PROMISE [58]	✓	0.95	0.94	0.96	0.95
[103]	Mining requirements from feedbacks on Q&A sites	Tokenization, stemming, stopword removal, remove special characters, SVM	PURE [188]	✓	-	0.72	0.77	0.74
[63]	Requirements extraction and classification	Hierarchical Cluster	Reviews from mobile apps		-	-	-	-
[41]	Mining requirements	Tokenization, POS Tagger, Rule Based	Traffic laws for Luxembourg, penalties for violating the inspection processes for vehicles, and interactions between the magistrates related to an ongoing prosecutions on traffic offenses.	-	0.97	0.94	-	
[47]	Classification of Business Rules	POS Tagging, Bi-LSTM	Publicly available SRS documents		0.73	0.697	-	-
[83]	Classify FR and NFR	Convolutional Nueral Network	PROMISE [58]	✓	-	0.80	0.78	0.77
[124]	Mining requirements	Lemmatization, stopword removal, remove special charcters, lowercasing, Aspect and Sentiment Unification Model (ASUM)	3 applications on Android Marketplace		0.90	0.80		
[73]	Glossary Term Extraction	Tokenization, lowercase, remove special characters, POS tagger, chunking, lemmatization	CrowdRE dataset [169]		-	0.73	0.84	-
[52]	Glossary Term Extraction	Expand contractionns, tokenization, POS Tag,text chunking,lowercase, Expectation Maximization (EM), FastText	Crowd RE dataset [169]	✓	-	0.84	0.53	0.65
[148]	Extracting requirements	BERT-QA	Technical texts from a Microelectromechanical Systems		-	0.8	0.72	-

TABLE 4. *(Continued.)* **Summary of all state-of-art techniques on the basis of three parameters, Accuracy (A), Precision (P), Recall (R), and F1-Score.**

[24]	Requirements Extraction	Tokenization, Lemmatization, stopword removal, remove special characters, POS Tagger, LSA	PROMISE	✓	-	0.75	0.59	-
[25]	Requirements Extraction	Tokenization, POS Tagger, Bi-LSTM	34 documents from students course projects	-	0.76	0.81	0.78	
[149]	Requirements Classification	Lemmatization, stopword removal, remove special characters, lowercasing, Multinomial Naive Bayes, Decision Tree, and Random Forest	4000 tweets		-	-	-	0.7
[37]	Mining requirements	Tokenization, lemmatization, stopword removal, POS Tagger, BERT	Industrial Cases and Public Dronology Dataset [190]		-	0.82	0.82	0.82
[66]	Identify duplicate requirements	BERT	PURE [188]	✓	0.75	-	-	0.87
[137]	Mining requirements	Tokenization, lemmatization, stopword removal, Naïve Bayes	Private dataset		0.91	-	-	-
[24]	Mining requirements	Lemmatization, stopword removal, POS Tagger, Rule based	PROMISE 'NFR' [191] and CCHIT [173]	✓	-	0.75	0.59	-

visualization can be used to illustrate the sequence of splits leading to a prediction. For more complex models, such as transformers, attention heatmaps [116] can be used to show which parts of the input data the model focused on when making a decision.

Seventh, it is imperative to conduct thorough root-cause analysis of errors incurred by ML-based methodologies. While studies reviewed typically rely on metrics such as precision, recall, accuracy, and/or F1-score to gauge the effectiveness of their approach, these metrics merely provide a binary assessment of model performance without delving into the underlying data issues that contribute to erroneous outputs. Consequently, there is a pressing need for future research to employ advanced methodologies capable of conducting comprehensive error analysis, thereby uncovering the precise data-related factors responsible for model inaccuracies. This nuanced understanding is essential for refining ML models and improving their reliability and effectiveness.

Lastly, future research endeavors should take into account the dynamic nature of user requirements. While the studies reviewed in this study, primarily utilized machine learning algorithms on a fixed set of requirements, it is crucial to recognize that stakeholder needs evolve over time in real-world scenarios. Therefore, there is a clear imperative for further investigation that concentrates on addressing the challenges posed by changing requirements. This could involve exploring methodologies that adapt to evolving requirements. Such research efforts would contribute significantly to enhancing the adaptability and effectiveness of machine learning-based approaches in requirements engineering.

VI. CONCLUSION

This literature review offers an overview of the current landscape regarding AI-driven approaches to requirements elicitation. Initially, we identified the diverse elicitation tasks prevalent in current research. Subsequently, we analysed NLP and M algorithms prominently featured in these studies. Finally, we thoroughly explored the publicly accessible corpora leveraged by the body of research.

Furthermore, our exploration revealed that while Naive Bayes and Support Vector Machines remain widely utilized algorithms, the pinnacle of performance is reached by the BERT language model as demonstrated in this assessment. Additionally, it is noteworthy that Precision, Recall, and F1 score emerge as the predominant evaluation metrics utilized to gauge model effectiveness.

Beyond these primary discoveries, a significant observation arises: the predominant focus of many articles centers on the classification of requirements into functional and non-functional categories. Moreover, supervised machine learning techniques dominate the landscape, with unsupervised methods such as clustering and Latent Dirichlet Allocation (LDA) also making notable appearances. Furthermore, the existing literature predominantly addresses fine-grained challenges in elicitation, such as discerning user reviews as indicative of requirements or not. Notably, absent from the reviewed research are the efforts to integrate AI-based solutions into existing elicitation workflows. Moreover, recent studies are deficient in comprehensive evaluation methodologies.

Lastly, it is important to highlight the potential role of recent advancements in Large Language Models (LLMs), which hold promise in assisting designers and engineers

to streamline requirements acquisition processes, thereby reducing time and effort expended in this domain.

APPENDIX.
LIST OF TABLES USED IN THE LITERATURE REVIEW

See Tables 3 and 4.

REFERENCES

[1] H. Meth, M. Brhel, and A. Maedche, "The state of the art in automated requirements elicitation," *Inf. Softw. Technol.*, vol. 55, no. 10, pp. 1695–1709, Oct. 2013.

[2] J. Manuel Pérez-Verdejo, A. J. Sánchez-García, and J. Octavio Ocharán-Hernández, "A systematic literature review on machine learning for automated requirements classification," in *Proc. 8th Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, Nov. 2020, pp. 21–28.

[3] D. A. López-Hernández, J. Octavio Ocharán-Hernández, E. Mezura-Montes, and Á. J. Sánchez-García, "Automatic classification of software requirements using artificial neural networks: A systematic literature review," in *Proc. 9th Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, Oct. 2021, pp. 152–160.

[4] A. Ahmad, C. Feng, M. Khan, A. Khan, A. Ullah, S. Nazir, and A. Tahir, "A systematic literature review on using machine learning algorithms for software requirements identification on stack overflow," *Secur. Commun. Netw.*, vol. 2020, pp. 1–19, Jul. 2020.

[5] M. Binkhonain and L. Zhao, "A review of machine learning algorithms for identification and classification of non-functional requirements," *Expert Syst. Appl., X*, vol. 1, Apr. 2019, Art. no. 100001.

[6] N. Handa, A. Sharma, and A. Gupta, "An inclusive study of several machine learning based non-functional requirements prediction techniques," in *Futuristic Trends in Networks and Computing Technologies* (Communications in Computer and Information Science), vol. 1206. Singapore: Springer, 2020, pp. 482–493.

[7] S. Lim, A. Henriksson, and J. Zdravkovic, "Data-driven requirements elicitation: A systematic literature review," *SN Comput. Sci.*, vol. 2, no. 1, p. 16, 2021.

[8] G. C. Sampada, T. I. Sake, and M. Chhabra, "A review on advanced techniques of requirement elicitation and specification in software development stages," in *Proc. 6th Int. Conf. Parallel, Distrib. Grid Comput. (PDGC)*, Nov. 2020, pp. 215–220.

[9] C. Cheligeer, J. Huang, G. Wu, N. Bhuiyan, Y. Xu, and Y. Zeng, "Machine learning in requirements elicitation: A literature review," *Artif. Intell. Eng. Des., Anal. Manuf.*, vol. 36, p. e32, Jan. 2022.

[10] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584908001390

[11] T. Iqbal, M. Khan, K. Taveter, and N. Seyff, "Mining Reddit as a new source for software requirements," in *Proc. IEEE 29th Int. Requirements Eng. Conf. (RE)*, Sep. 2021, pp. 128–138.

[12] J. A. Khan, Y. Xie, L. Liu, and L. Wen, "Analysis of requirements-related arguments in user forums," in *Proc. IEEE 27th Int. Requirements Eng. Conf. (RE)*, Sep. 2019, pp. 63–74.

[13] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy, "NoRBERT: Transfer learning for requirements classification," in *Proc. IEEE 28th Int. Requirements Eng. Conf. (RE)*, Aug. 2020, pp. 169–179.

[14] N. Munaiah, A. Meneely, and P. K. Murukannaiah, "A domain-independent model for identifying security requirements," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 506–511.

[15] D. Siahaan, I. K. Raharjana, and C. Fatichah, "User story extraction from natural language for requirements elicitation: Identify software-related information from online news," *Inf. Softw. Technol.*, vol. 158, Jun. 2023, Art. no. 107195.

[16] A. M. Alashqar, "Studying the commonalities, mappings and relationships between non-functional requirements using machine learning," *Sci. Comput. Program.*, vol. 218, Jun. 2022, Art. no. 102806. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642322000399

[17] I. K. Raharjana, D. Siahaan, and C. Fatichah, "User story extraction from online news for software requirements elicitation: A conceptual model," in *Proc. 16th Int. Joint Conf. Comput. Sci. Softw. Eng. (JCSSE)*, Jul. 2019, pp. 342–347.

[18] M. Rahmi Dewi, I. Kharisma Raharjana, D. Siahaan, and C. Fatichah, "Software requirement-related information extraction from online news using domain specificity for requirements elicitation," in *Proc. 10th Int. Conf. Softw. Comput. Appl.*, 2021, pp. 81–87.

[19] S. Mishra and A. Sharma, "A generalized semantic filter for glossary term extraction from large-sized software requirements," in *Proc. 14th Innov. Softw. Eng. Conf. (Formerly Known India Softw. Eng. Conf.)*, 2021, pp. 1–9.

[20] M. Hamza and R. J. Walker, "Recommending features and feature relationships from requirements documents for software product lines," in *Proc. IEEE/ACM 4th Int. Workshop Realizing Artif. Intell. Synergies Softw. Eng.*, May 2015, pp. 25–31.

[21] M. Dragoni, M. Federici, and A. Rexha, "An unsupervised aspect extraction strategy for monitoring real-time reviews stream," *Inf. Process. Manage.*, vol. 56, no. 3, pp. 1103–1118, May 2019.

[22] T. Johann, C. Stanik, A. M. Alizadeh, and W. Maalej, "SAFE: A simple approach for feature extraction from app descriptions and app reviews," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 21–30.

[23] A. Sleimi, M. Ceci, M. Sabetzadeh, L. C. Briand, and J. Dann, "Automated recommendation of templates for legal requirements," in *Proc. IEEE 28th Int. Requirements Eng. Conf. (RE)*, Aug. 2020, pp. 158–168.

[24] M. Younas, D. N. A. Jawawi, I. Ghani, and M. A. Shah, "Extraction of non-functional requirement using semantic similarity distance," *Neural Comput. Appl.*, vol. 32, no. 11, pp. 7383–7397, Jun. 2020.

[25] Y. Wang and J. Zhang, "Experiment on automatic functional requirements analysis with the EFRF's semantic cases," in *Proc. Int. Conf. Prog. Informat. Comput. (PIC)*, Dec. 2016, pp. 636–642.

[26] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Extracting domain models from natural-language requirements: Approach and industrial evaluation," in *Proc. ACM/IEEE 19th Int. Conf. Model Driven Eng. Lang. Syst.*, New York, NY, USA, 2016, pp. 250–260. [Online]. Available: https://doi-org.uproxy.library.dc-uoit.ca/10.1145/2976767.2976769

[27] S. Abualhaija, C. Arora, M. Sabetzadeh, L. C. Briand, and E. Vaz, "A machine learning-based approach for demarcating requirements in textual specifications," in *Proc. IEEE 27th Int. Requirements Eng. Conf. (RE)*, Sep. 2019, pp. 51–62.

[28] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Requirements classification with interpretable machine learning and dependency parsing," in *Proc. IEEE 27th Int. Requirements Eng. Conf. (RE)*, Sep. 2019, pp. 142–152.

[29] Y. Ko, S. Park, J. Seo, and S. Choi, "Using classification techniques for informal requirements in the requirements analysis-supporting system," *Inf. Softw. Technol.*, vol. 49, nos. 11–12, pp. 1128–1140, Nov. 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584906001947

[30] M. Mezghani, J. Kang, and F. Sèdes, "Using k-means for redundancy and inconsistency detection: Application to industrial requirements," in *Proc. HAL*. Cham, Switzerland: Springer, 2018, pp. 501–508.

[31] R. Sonbol, G. Rebdawi, and N. Ghneim, "Learning software requirements syntax: An unsupervised approach to recognize templates," *Knowl.-Based Syst.*, vol. 248, Jul. 2022, Art. no. 108933. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705122004518

[32] F. Casillo, V. Deufemia, and C. Gravino, "Detecting privacy requirements from user stories with NLP transfer learning models," *Inf. Softw. Technol.*, vol. 146, Jun. 2022, Art. no. 106853. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584922000246

[33] A. Al-Hroob, A. T. Imam, and R. Al-Heisa, "The use of artificial neural networks for extracting actions and actors from requirements document," *Inf. Softw. Technol.*, vol. 101, pp. 1–15, Sep. 2018.

[34] Q. A. Do, T. Bhowmik, and G. L. Bradshaw, "Capturing creative requirements via requirements reuse: A machine learning-based approach," *J. Syst. Softw.*, vol. 170, Dec. 2020, Art. no. 110730. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121220301631

[35] M. R. Asadabadi, M. Saberi, O. Zwikael, and E. Chang, "Ambiguous requirements: A semi-automated approach to identify and clarify ambiguity in large-scale projects," *Comput. Ind. Eng.*, vol. 149, Nov. 2020, Art. no. 106828.

[36] N. H. Bakar, Z. M. Kasirun, N. Salleh, and H. A. Jalab, "Extracting features from online software reviews to aid requirements reuse," *Appl. Soft Comput.*, vol. 49, pp. 1297–1315, Dec. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494616303830

[37] S. Bashir, M. Abbas, M. Saadatmand, E. P. Enoiu, M. Bohlin, and P. Lindberg, "Requirement or not, that is the question: A case from the railway industry," in *Requirements Engineering: Foundation for Software Quality*, A. Ferrari and B. Penzenstadler, Eds. Cham, Switzerland: Springer, 2023, pp. 105–121.

[38] J. Jeong and N. Kim, "Does sentiment help requirement engineering: Exploring sentiments in user comments to discover informative comments," *Automated Softw. Eng.*, vol. 28, no. 2, p. 18, Nov. 2021, doi: 10.1007/s10515-021-00295-w.

[39] I. S. Bajwa, T. Sibalija, and D. N. A. Jawawi, "An intelligent approach for CRC models based agile software requirement engineering using SBVR," in *Intelligent Technologies and Applications* (Communications in Computer and Information Science), vol. 1198. Singapore: Springer, 2020, pp. 372–384.

[40] Z. Kurtanovic and W. Maalej, "Mining user rationale from software reviews," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 61–70.

[41] A. Sleimi, N. Sannier, M. Sabetzadeh, L. Briand, and J. Dann, "Automated extraction of semantic legal metadata using natural language processing," in *Proc. IEEE 26th Int. Requirements Eng. Conf. (RE)*, Aug. 2018, pp. 124–135.

[42] M. Mezghani, J. Kang, and F. Sèdes, "Industrial requirements classification for redundancy and inconsistency detection in SEMIOS," in *Proc. IEEE 26th Int. Requirements Eng. Conf. (RE)*, Aug. 2018, pp. 297–303.

[43] T. Gemkow, M. Conzelmann, K. Hartig, and A. Vogelsang, "Automatic glossary term extraction from large-scale requirements specifications," in *Proc. IEEE 26th Int. Requirements Eng. Conf. (RE)*, Aug. 2018, pp. 412–417.

[44] J. Zhang, S. Chen, J. Hua, N. Niu, and C. Liu, "Automatic terminology extraction and ranking for feature modeling," in *Proc. IEEE 30th Int. Requirements Eng. Conf. (RE)*, Aug. 2022, pp. 51–63.

[45] S. Tiwari, S. S. Rathore, S. Sagar, and Y. Mirani, "Identifying use case elements from textual specification: A preliminary study," in *Proc. IEEE 28th Int. Requirements Eng. Conf. (RE)*, Aug. 2020, pp. 410–411.

[46] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *Proc. IEEE 22nd Int. Requirements Eng. Conf. (RE)*, Aug. 2014, pp. 153–162.

[47] P. R. Anish, P. Lawhatre, R. Chatterjee, V. Joshi, and S. Ghaisas, "Automated labeling and classification of Bus. Rules from software requirement specifications," in *Proc. IEEE/ACM 44th Int. Conf. Softw. Eng., Softw. Eng. Pract. (ICSE-SEIP)*, May 2022, pp. 53–54.

[48] T. Panthum and T. Senivongse, "Generating functional requirements based on classification of mobile application user reviews," in *Proc. IEEE/ACIS 19th Int. Conf. Softw. Eng. Res., Manage. Appl. (SERA)*, Jun. 2021, pp. 15–20.

[49] V. Nama, G. Deepak, and A. Santhanavijayan, "KCReqRec: A knowledge centric approach for semantically inclined requirement recommendation with micro requirement mapping using hybrid learning models," in *Intelligent Systems Design and Applications*, A. Abraham, S. Pllana, G. Casalino, K. Ma, and A. Bajaj, Eds. Cham, Switzerland: Springer, 2023, pp. 12–22.

[50] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Improving requirements glossary construction via clustering: Approach and industrial case studies," in *Proc. 8th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas.*, 2014, pp. 1–10.

[51] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated extraction and clustering of requirements glossary terms," *IEEE Trans. Softw. Eng.*, vol. 43, no. 10, pp. 918–945, Oct. 2017.

[52] K. Bhatia, S. Mishra, and A. Sharma, "Clustering glossary terms extracted from large-sized software requirements using FastText," in *Proc. 13th Innov. Softw. Eng. Conf. (Formerly Known India Softw. Eng. Conf.)*, 2020, pp. 1–11.

[53] D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 740–750. [Online]. Available: https://aclanthology.org/D14-1082

[54] N. Kengphanphanit and P. Muenchaisri, "Automatic requirements elicitation from social media (ARESM)," in *Proc. Int. Conf. Comput. Commun. Inf. Syst.*, 2020, pp. 57–62.

[55] H. Villamizar, M. Kalinowski, A. Garcia, and D. Mendez, "An efficient approach for reviewing security-related aspects in agile requirements specifications of web applications," *Requirements Eng.*, vol. 25, no. 4, pp. 439–468, Dec. 2020.

[56] H. Yang and P. Liang, "Identification and classification of requirements from app user reviews," in *Proc. SEKE*, 2015, pp. 7–12.

[57] G. Y. Quba, H. Al Qaisi, A. Althunibat, and S. AlZu'bi, "Software requirements classification using machine learning algorithm's," in *Proc. Int. Conf. Inf. Technol. (ICIT)*, Jul. 2021, pp. 685–690.

[58] J. Cleland-Huang, S. Mazrouee, H. Liguo, and D. Port, "NFR," Zenodo, Feb. 2017, doi: 10.5281/zenodo.268542.

[59] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Eng.*, vol. 21, no. 3, pp. 311–331, Sep. 2016.

[60] V. T. Dhinakaran, R. Pulle, N. Ajmeri, and P. K. Murukannaiah, "App review analysis via active learning: Reducing supervision effort without compromising classification accuracy," in *Proc. IEEE 26th Int. Requirements Eng. Conf. (RE)*, Aug. 2018, pp. 170–181.

[61] M. Asif, I. Ali, M. S. A. Malik, M. H. Chaudary, S. Tayyaba, and M. T. Mahmood, "Annotation of software requirements specification (SRS), extractions of nonfunctional requirements, and measurement of their tradeoff," *IEEE Access*, vol. 7, pp. 36164–36176, 2019.

[62] E. Guzman, M. Ibrahim, and M. Glinz, "A little bird told me: Mining tweets for requirements and software evolution," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 11–20.

[63] A. Araujo and R. Marcacini, "Hierarchical cluster labeling of software requirements using contextual word embeddings," in *Proc. XXXV Brazilian Symp. Softw. Eng.*, 2021, pp. 297–302.

[64] S. Abualhaija, C. Arora, A. Sleimi, and L. C. Briand, "Automated question answering for improved understanding of compliance requirements: A multi-document study," in *Proc. IEEE 30th Int. Requirements Eng. Conf. (RE)*, Aug. 2022, pp. 39–50.

[65] C. Li, L. Huang, J. Ge, B. Luo, and V. Ng, "Automatically classifying user requests in crowdsourcing requirements engineering," *J. Syst. Softw.*, vol. 138, pp. 108–123, Apr. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121217303096

[66] D. Rao, L. Bian, and H. Zhao, "Research of duplicate requirement detection method," in *Smart Computing and Communication*, M. Qiu, Z. Lu, and C. Zhang, Eds. Cham, Switzerland: Springer, 2023, pp. 213–225.

[67] L. Yi, W. Zhang, H. Zhao, Z. Jin, and H. Mei, "Mining binary constraints in the construction of feature models," in *Proc. 20th IEEE Int. Requirements Eng. Conf. (RE)*, Sep. 2012, pp. 141–150.

[68] A. Dekhtyar and V. Fong, "RE data challenge: Requirements identification with Word2 Vec and TensorFlow," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 484–489.

[69] X. Liu, Y. Leng, W. Yang, C. Zhai, and T. Xie, "Mining Android app descriptions for permission requirements recommendation," in *Proc. IEEE 26th Int. Requirements Eng. Conf. (RE)*, Aug. 2018, pp. 147–158.

[70] E. Guzman, R. Alkadhi, and N. Seyff, "A needle in a haystack: What do Twitter users say about software?" in *Proc. IEEE 24th Int. Requirements Eng. Conf. (RE)*, Sep. 2016, pp. 96–105.

[71] M.-I. Limaylla-Lunarejo, N. Condori-Fernandez, and M. R. Luaces, "Towards an automatic requirements classification in a new Spanish dataset," in *Proc. IEEE 30th Int. Requirements Eng. Conf. (RE)*, Aug. 2022, pp. 270–271.

[72] M. Lu and P. Liang, "Automatic classification of non-functional requirements from augmented app user reviews," in *Proc. 21st Int. Conf. Eval. Assessment Softw. Eng.*, 2017, pp. 344–353.

[73] S. Mishra and A. Sharma, "Automatic word embeddings-based glossary term extraction from large-sized software requirements," in *Requirements Engineering: Foundation for Software Quality* (Lecture Notes in Computer Science). Springer, 2020, ch. 15, pp. 203–218.

[74] J. Winkler and A. Vogelsang, "Automatic classification of requirements based on convolutional neural networks," in *Proc. IEEE 24th Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2016, pp. 39–45.

[75] K. Jeon, G. Lee, and H. D. Jeong, "Classification of the requirement sentences of the U,S. DOT standard specification using deep learning algorithms," in *Proc. 18th Int. Conf. Comput. Civil Building Eng.* (Lecture Notes in Civil Engineering). Cham, Switzerland: Springer, 2021, pp. 89–97.

[76] L. Kumar, S. Baldwa, S. M. Jambavalikar, L. B. Murthy, and A. Krishna, "Software functional and non-function requirement classification using word-embedding," in *Advanced Information Networking and Applications*, L. Barolli, F. Hussain, and T. Enokido, Eds. Cham, Switzerland: Springer, 2022, pp. 167–179.

[77] M. B. Hosseini, J. Heaps, R. Slavin, J. Niu, and T. Breaux, "Ambiguity and generality in natural language privacy policies," in *Proc. IEEE 29th Int. Requirements Eng. Conf. (RE)*, Sep. 2021, pp. 70–81.

[78] A. F. de Araújo and R. M. Marcacini, "RE-BERT: Automatic extraction of software requirements from app reviews using BERT language model," in *Proc. 36th Annu. ACM Symp. Appl. Comput.*, 2021, pp. 1321–1327. [Online]. Available: https://doi-org.uproxy.library.dc-uoit.ca/10.1145/3412841.3442006

[79] S. Abualhaija, C. Arora, and L. C. Briand, "COREQQA: A COmpliance REQuirements understanding using question answering tool," in *Proc. 30th ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2022, pp. 1682–1686. [Online]. Available: https://doi-org.uproxy.library.dc-uoit.ca/10.1145/3540250.3558926

[80] D. Kici, A. Bozanta, M. Cevik, D. Parikh, and A. Basar, "Text classification on software requirements specifications using transformer models," in *Proc. 31st Annu. Int. Conf. Comput. Sci. Softw. Eng.*, 2021, pp. 163–172.

[81] M. Li, L. Shi, Y. Yang, and Q. Wang, "A deep multitask learning approach for requirements discovery and annotation from open forum," in *Proc. 35th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2020, pp. 336–348.

[82] A. Araujo, M. Golo, B. Viana, F. Sanches, R. Romero, and R. Marcacini, "From bag-of-words to pre-trained neural language models: Improving automatic classification of app reviews for requirements engineering," in *Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional*. Porto Alegre, Brasil: SBC, 2020, pp. 378–389, doi: 10.5753/eniac.2020.12144. [Online]. Available: https://sol.sbc.org.br/index.php/eniac/article/view/12144

[83] R. Navarro-Almanza, R. Juarez-Ramirez, and G. Licea, "Towards supporting software engineering using deep learning: A case of software requirements classification," in *Proc. 5th Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, Oct. 2017, pp. 116–120.

[84] K. Kaur and P. Kaur, "SABDM: A self-attention based bidirectional-RNN deep model for requirements classification," *J. Softw., Evol. Process*, vol. 36, no. 2, p. e2430, 2022.

[85] Z. Kurtanovic and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 490–495.

[86] J. Tizard, H. Wang, L. Yohannes, and K. Blincoe, "Can a conversation paint a picture? Mining requirements in software forums," in *Proc. IEEE 27th Int. Requirements Eng. Conf. (RE)*, Sep. 2019, pp. 17–27.

[87] P. Devine, J. Tizard, H. Wang, Y. S. Koh, and K. Blincoe, "What's inside a cluster of software user feedback: A study of characterisation methods," in *Proc. IEEE 30th Int. Requirements Eng. Conf. (RE)*, Aug. 2022, pp. 189–200.

[88] M. Binkhonain and L. Zhao, "A machine learning approach for hierarchical classification of software requirements," *Mach. Learn. Appl.*, vol. 12, Jun. 2023, Art. no. 100457. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666827023000105

[89] P. Kadebu, S. Sikka, R. K. Tyagi, and P. Chiurunge, "A classification approach for software requirements towards maintainable security," *Sci. Afr.*, vol. 19, Mar. 2023, Art. no. e01496. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468227622004008

[90] J. H. Hayes, W. Li, and M. Rahimi, "Weka meets TraceLab: Toward convenient classification: Machine learning for requirements engineering problems: A position paper," in *Proc. IEEE 1st Int. Workshop Artif. Intell. Requirements Eng. (AIRE)*, Aug. 2014, pp. 9–12.

[91] T. Bhowmik, N. Niu, A. Mahmoud, and J. Savolainen, "Automated support for combinational creativity in requirements engineering," in *Proc. IEEE 22nd Int. Requirements Eng. Conf. (RE)*, Aug. 2014, pp. 243–252.

[92] H. Jiang, J. Zhang, X. Li, Z. Ren, D. Lo, X. Wu, and Z. Luo, "Recommending new features from mobile app descriptions," *ACM Trans. Softw. Eng. Methodol.*, vol. 28, no. 4, pp. 1–29, Oct. 2019.

[93] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider, "What works better? A study of classifying requirements," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 496–501.

[94] Y. Li, E. Guzman, K. Tsiamoura, F. Schneider, and B. Bruegge, "Automated requirements extraction for scientific software," *Proc. Comput. Sci.*, vol. 51, pp. 582–591, Jan. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915011345

[95] Q. Y. Shambour, A. H. Hussein, Q. M. Kharma, and M. M. Abualhaj, "Effective hybrid content-based collaborative filtering approach for requirements engineering," *Comput. Syst. Sci. Eng.*, vol. 40, no. 1, pp. 113–125, 2022. [Online]. Available: http://www.techscience.com/csse/v40n1/44216

[96] D. Adanza Dopazo, V. Moreno Pelayo, and G. Génova Fuster, "An automatic methodology for the quality enhancement of requirements using genetic algorithms," *Inf. Softw. Technol.*, vol. 140, Dec. 2021, Art. no. 106696. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584921001518

[97] H. Akay and S.-G. Kim, "Reading functional requirements using machine learning-based language processing," *CIRP Ann.*, vol. 70, no. 1, pp. 139–142, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0007850621000457

[98] Z. Chen, I. Li, H. Zhang, S. Preum, J. A. Stankovic, and M. Ma, "CitySpec with shield: A secure intelligent assistant for requirement formalization," *Pervas. Mobile Comput.*, vol. 92, May 2023, Art. no. 101802. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574119223000603

[99] F. Sarro, M. Harman, Y. Jia, and Y. Zhang, "Customer rating reactions can be predicted purely using app features," in *Proc. IEEE 26th Int. Requirements Eng. Conf. (RE)*, Aug. 2018, pp. 76–87.

[100] B. Li, Z. Li, and Y. Yang, "NFRNet: A deep neural network for automatic classification of non-functional requirements," in *Proc. IEEE 29th Int. Requirements Eng. Conf. (RE)*, Sep. 2021, pp. 434–435.

[101] Y. Wang, L. Zheng, and N. Li, "ROM: A requirement opinions mining method preliminary try based on software review data," in *Proc. 4th Int. Conf. Manage. Eng., Softw. Eng. Service Sci.*, 2020, pp. 26–33.

[102] I. Morales-Ramirez, F. M. Kifetew, and A. Perini, "Analysis of online discussions in support of requirements discovery," in *Proc. 29th Int. Conf. Adv. Inf. Syst. Eng.*, 2017, pp. 159–174.

[103] L. Liu and M. Aoyama, "Requirement acquisition from social Q&A sites," in *Proc. APRES* (Communications in Computer and Information Science), vol. 558. Berlin, Germany:Springer, 2015, pp. 64–74.

[104] *Common Weakness Enumeration*. [Online]. Available: https://cwe.mitre.org/about/index.html

[105] P. Rodeghero, S. Jiang, A. Armaly, and C. McMillan, "Detecting user story information in developer-client conversations to generate extractive summaries," in *Proc. IEEE/ACM 39th Int. Conf. Softw. Eng. (ICSE)*, May 2017, pp. 49–59.

[106] C. Stanik, M. Haering, C. Jesdabodi, and W. Maalej, "Which app features are being used? Learning app feature usages from interaction data," in *Proc. IEEE 28th Int. Requirements Eng. Conf. (RE)*, Aug. 2020, pp. 66–77.

[107] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," 2017, *arXiv:1711.10160*.

[108] R. Chatterjee, A. Ahmed, P. Rose Anish, B. Suman, P. Lawhatre, and S. Ghaisas, "A pipeline for automating labeling to prediction in classification of NFRs," in *Proc. IEEE 29th Int. Requirements Eng. Conf. (RE)*, Sep. 2021, p. 323.

[109] K. Kaur and P. Kaur, "Improving BERT model for requirements classification by bidirectional LSTM-CNN deep model," *Comput. Electr. Eng.*, vol. 108, May 2023, Art. no. 108699. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790623001234

[110] C. El Morr, M. Jammal, H. Ali-Hassan, and W. El-Hallak, *Support Vector Machine*. Cham, Switzerland: Springer, 2022, pp. 385–411, doi: 10.1007/978-3-031-16990-8_13.

[111] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *Social Netw. Comput. Sci.*, vol. 2, no. 3, p. 160, May 2021, doi: 10.1007/s42979-021-00592-x.

[112] I. H. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *Social Netw. Comput. Sci.*, vol. 2, no. 6, p. 420, Aug. 2021, doi: 10.1007/s42979-021-00815-1.

[113] T. Li and Z. Chen, "An ontology-based learning approach for automatically classifying security requirements," *J. Syst. Softw.*, vol. 165, Jul. 2020, Art. no. 110566.

[114] K. E. Wiegers and K. E. Wiegers, *Software Requirements*. Washington, DC, USA: Microsoft Press, 2005.

[115] R. Indraswari, R. Rokhana, and W. Herulambang, "Melanoma image classification based on MobileNetV2 network," *Proc. Comput. Sci.*, vol. 197, pp. 198–207, Jan. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921023565

[116] J. Vig, "A multiscale visualization of attention in the transformer model," 2019, *arXiv:1906.05714*.

[117] P. R. Anish, B. Balasubramaniam, A. Sainani, J. Cleland-Huang, M. Daneva, R. J. Wieringa, and S. Ghaisas, "Probing for requirements knowledge to stimulate architectural thinking," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng. (ICSE)*, May 2016, pp. 843–854.

[118] G. Williams and A. Mahmoud, "Mining Twitter feeds for software user requirements," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 1–10.

[119] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," in *Proc. IEEE 23rd Int. Requirements Eng. Conf. (RE)*, Aug. 2015, pp. 116–125.

[120] O. AlDhafer, I. Ahmad, and S. Mahmood, "An end-to-end deep learning system for requirements classification using recurrent neural networks," *Inf. Softw. Technol.*, vol. 147, Jul. 2022, Art. no. 106877.

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

[122] C. Baker, L. Deng, S. Chakraborty, and J. Dehlinger, "Automatic multi-class non-functional software requirements classification using neural networks," in *Proc. IEEE 43rd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jul. 2019, pp. 610–615.

[123] E. Guzman, M. El-Haliby, and B. Bruegge, "Ensemble methods for app review classification: An approach for software evolution (N)," in *Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2015, pp. 771–776.

[124] L. V. G. Carreño and K. Winbladh, "Analysis of user comments: An approach for software requirements evolution," in *Proc. 35th Int. Conf. Softw. Eng. (ICSE)*, May 2013, pp. 582–591.

[125] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang, "The app sampling problem for app store mining," in *Proc. IEEE/ACM 12th Work. Conf. Mining Softw. Repositories*, May 2015, pp. 123–133.

[126] C. Wang, F. Zhang, P. Liang, M. Daneva, and M. van Sinderen, "Can app changelogs improve requirements classification from app reviews?: An exploratory study," in *Proc. 12th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas.*, 2018, pp. 1–4.

[127] H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Speculative requirements: Automatic detection of uncertainty in natural language requirements," in *Proc. 20th IEEE Int. Requirements Eng. Conf. (RE)*, Sep. 2012, pp. 11–20.

[128] T. Merten, M. Falis, P. Hübner, T. Quirchmayr, S. Bürsner, and B. Paech, "Software feature request detection in issue tracking systems," in *Proc. IEEE 24th Int. Requirements Eng. Conf. (RE)*, Sep. 2016, pp. 166–175.

[129] M. N. Al Islam, Y. Ma, P. Alarcon, N. Chawla, and J. Cleland-Huang, "RESAM: Requirements elicitation and specification for deep-learning anomaly models with applications to UAV flight controllers," in *Proc. IEEE 30th Int. Requirements Eng. Conf. (RE)*, Aug. 2022, pp. 153–165.

[130] C. Stanik, T. Pietz, and W. Maalej, "Unsupervised topic discovery in user comments," in *Proc. IEEE 29th Int. Requirements Eng. Conf. (RE)*, Sep. 2021, pp. 150–161.

[131] I. P. Delgado-Solano, A. S. Núñez-Varela, and G. Héctor Pérez-González, "Keyword extraction from Users' requirements using TextRank and frequency analysis, and their classification into ISO/IEC 25000 quality categories," in *Proc. 6th Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, Oct. 2018, pp. 88–92.

[132] A. Ciurumelea, A. Schaufelbühl, S. Panichella, and H. C. Gall, "Analyzing reviews and code of mobile apps for better release planning," in *Proc. IEEE 24th Int. Conf. Softw. Anal., Evol. Reeng. (SANER)*, Feb. 2017, pp. 91–102.

[133] A. Sainani, P. R. Anish, V. Joshi, and S. Ghaisas, "Extracting and classifying requirements from software engineering contracts," in *Proc. IEEE 28th Int. Requirements Eng. Conf. (RE)*, Aug. 2020, pp. 147–157.

[134] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "AR-miner: Mining informative reviews for developers from mobile app marketplace," in *Proc. 36th Int. Conf. Softw. Eng.*, May 2014, pp. 767–778.

[135] J. Fischbach, J. Frattini, A. Spaans, M. Kummeth, A. Vogelsang, D. Mendez, and M. Unterkalmsteiner, "Automatic detection of causality in requirement artifacts: The cira approach," in *Requirements Engineering: Foundation for Software Quality*, F. Dalpiaz and P. Spoletini, Eds. Cham, Switzerland: Springer, 2021, pp. 19–36.

[136] D. Ott, "Automatic requirement categorization of large natural language specifications at mercedes-benz for review improvements," in *Requirements Engineering: Foundation for Software Quality*, J. Doerr and A. L. Opdahl, Eds. Berlin, Germany: Springer, 2013, pp. 50–64.

[137] T. Le, C. Le, H. David Jeong, S. B. Gilbert, and E. Chukharev-Hudilainen, "Requirement text detection from contract packages to support project definition determination," in *Advances in Informatics and Computing in Civil and Construction Engineering*, I. Mutis and T. Hartmann, Eds. Cham, Switzerland: Springer, 2019, pp. 569–576.

[138] M. van Vliet, E. C. Groen, F. Dalpiaz, and S. Brinkkemper, "Identifying and classifying user requirements in online feedback via crowdsourcing," in *Requirements Engineering: Foundation for Software Quality*, N. Madhavji, L. Pasquale, A. Ferrari, and S. Gnesi, Eds. Cham, Switzerland: Springer, 2020, pp. 143–159.

[139] Q. A. Do and T. Bhowmik, "Automated generation of creative software requirements: A data-driven approach," in *Proc. 1st ACM SIGSOFT Int. Workshop Automated Specification Inference*, 2018, pp. 9–12.

[140] R. R. R. Merugu and S. R. Chinnam, "Automated cloud service based quality requirement classification for software requirement specification," *Evol. Intell.*, vol. 14, no. 2, pp. 389–394, Jun. 2021, doi: 10.1007/s12065-019-00241-6.

[141] S. Dustdar, E. Yu, C. Salinesi, D. Rieu, and V. Pant, "Mining user opinions to support requirement engineering: An empirical study," in *Advanced Information Systems Engineering* (Lecture Notes in Computer Science). vol. 12127, Cham, Switzerland: Springer, 2020, pp. 401–416.

[142] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "Automated classification of non-functional requirements," *Requirements Eng.*, vol. 12, no. 2, pp. 103–120, Apr. 2007.

[143] R. R. Mekala, A. Irfan, E. C. Groen, A. Porter, and M. Lindvall, "Classifying user requirements from online feedback in small dataset environments using deep learning," in *Proc. IEEE 29th Int. Requirements Eng. Conf. (RE)*, Sep. 2021, pp. 139–149.

[144] X. Wang and N. El-Gohary, "Deep learning-based relation extraction and knowledge graph-based representation of construction safety requirements," *Autom. Construction*, vol. 147, Mar. 2023, Art. no. 104696.

[145] B. Li and X. Nong, "Automatically classifying non-functional requirements using deep neural network," *Pattern Recognit.*, vol. 132, Dec. 2022, Art. no. 108948. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320322004289

[146] W. Alhoshan, A. Ferrari, and L. Zhao, "Zero-shot learning for requirements classification: An exploratory study," *Inf. Softw. Technol.*, vol. 159, Jul. 2023, Art. no. 107202. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584923000563

[147] C. Magalhães, J. Araujo, and A. Sardinha, "MARE: An active learning approach for requirements classification," in *Proc. IEEE 29th Int. Requirements Eng. Conf. (RE)*, Sep. 2021, pp. 516–521.

[148] H. Akay, M. Yang, and S.-G. Kim, "Automating design requirement extraction from text with deep learning," in *Proc. ASME Int. Design Eng. Tech. Conf. Comput. Inf. Eng. Conf. (IDETC-CIE), 47th Design Automat. Conf. (DAC)*, 2021, pp. 1–11.

[149] A. M. Ebrahimi and A. A. Barforoush, "Preprocessing role in analyzing tweets towards requirement engineering," in *Proc. 27th Iranian Conf. Electr. Eng. (ICEE)*, Apr. 2019, pp. 1905–1911.

[150] S. Scalabrino, G. Bavota, B. Russo, M. D. Penta, and R. Oliveto, "Listening to the crowd for the release planning of mobile apps," *IEEE Trans. Softw. Eng.*, vol. 45, no. 1, pp. 68–86, Jan. 2019.

[151] M. Riaz, J. King, J. Slankas, and L. Williams, "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts," in *Proc. IEEE 22nd Int. Requirements Eng. Conf. (RE)*, Aug. 2014, pp. 183–192.

[152] P. Gr?nbacher and A. Perini, "Mining user requirements from application store reviews using frame semantics," in *Proc. REFSQ* (Lecture Notes in Computer Science). vol. 10153, Cham, Switzerland: Springer, 2017, pp. 273–287.

[153] (2015). *'The Promise Repository of Empirical Software Engineering Data'*. [Online]. Available: https://zenodo.org/record/268452#.Wp8O4uejnIU

[154] *Real-Time Transit Information System Requirements*. Accessed: Dec. 30, 2023. [Online]. Available: https://mtc.ca.gov/planning/tcip/Real-Time_TransitSystemRequirements_v3.0.pdf

[155] (2020). *OSHA*. [Online]. Available: https://www.osha.gov/laws-regs/regulations/standardnumber/1926

[156] M. Lima, V. Valle, E. Costa, F. Lira, and B. Gadelha, "Software engineering repositories: Expanding the promise database," in *Proc. XXXIII Brazilian Symp. Softw. Eng.*, New York, NY, USA, 2019, pp. 427–436, doi: 10.1145/3350768.3350776.

[157] M. Data. (2022). *Dosspre*. [Online]. Available: https://data.mendeley.com/datasets/23xtbvk6yp/19
[158] D. Pearce and H.-G. Hirsch, "The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *Proc. Autom. Speech Recognit., Challenges New Millenium ISCA Tutorial Res. Workshop*, Oct. 2000, pp. 29–32.
[159] R. Callan, *Building Object-Oriented Systems: An Introduction From Concepts to Implementation in C++*. Southampton, U.K.: Computational Mechanics Publications, 1994.
[160] (Nov. 2019). *Manually Annotated Dataset and an Annotation Guideline for Caise 2020 Paper*. [Online]. Available: https://github.com/jsdabrowski/CAiSE-20/
[161] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of Google play," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 1, pp. 221–233, Jun. 2014, doi: 10.1145/2637364.2592003.
[162] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, Nov. 2009, doi: 10.1145/1656274.1656278.
[163] *Owasp: The Open Web Application Security Project*. Accessed: Mar. 3, 2024. [Online]. Available: https://owasp.org
[164] N. R. C. of Italy. *Natural Language Requirement Dataset, SRS Comparison*. Accessed: Mar. 3, 2024. [Online]. Available: http://fmt.isti.cnr.it/nlreqdataset
[165] *REUTS*. Accessed: Mar. 3, 2024. [Online]. Available: http://research.it.uts.edu.au/re/
[166] *UCLappA*. Accessed: Mar. 3, 2024. [Online]. Available: https://sites.google.com/site/saurabhiiitdmj/resources/repository.zip
[167] *UCL App Store Analysis Group*. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/F.Sarro/projects/UCLappA/home.html
[168] W. Maalej and H. Nabil. *On the automatic classification of App Reviews: Project Data*. https://mast.informatik.uni-hamburg.de/app-review-analysis/
[169] P. K. Murukannaiah, N. Ajmeri, and M. P. Singh, "Acquiring creative requirements from the crowd: Understanding the influences of personality and creative potential in crowd RE," in *Proc. IEEE 24th Int. Requirements Eng. Conf. (RE)*, Sep. 2016, pp. 176–185.
[170] P. K. Murukannaiah, N. Ajmeri, and M. P. Singh, "Toward automating crowd RE," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 512–515.
[171] M.-I. Limaylla-Lunarejo, N. Condori-Fernandez, and M. R. Luaces, "Towards an automatic requirements classification in a new Spanish dataset," in *Proc. IEEE 30th Int. Requirements Eng. Conf. (RE)*. USA: IEEE Computer Society, Oct. 2022, pp. 270–271, doi: 10.1109/RE54965.2022.00039.
[172] *Uni-Hannover*. Accessed: Mar. 3, 2024. [Online]. Available: http://www.se.uni-hannover.de/pages/en:projekte_re_secreq
[173] CCHIT. (2005). *Cchit Ambulatory Requirements*. [Online]. Available: https://cchit.org
[174] Accessed: Mar. 3, 2024. [Online]. Available: http://www.hl7.org/
[175] [Online]. Available: https://www.infoway-inforoute.ca/
[176] [Online]. Available: http://oscarcanada.org/
[177] [Online]. Available: http://www.va.gov/vler
[178] *'Software Requirements Specification'*. Accessed: Mar. 3, 2024. [Online]. Available: http://users.csc.calpoly.edu/~jdalbey/SWE/QA/nonfunctional.html
[179] *'Software Requirements Specification'*. Accessed: Mar. 3, 2024. [Online]. Available: https://requirementsquest.com/wpcontent/uploads/2017/01/Nonfunctional-RequirementEXAMPLES.pdf
[180] *'Software Requirements Specification'*. Accessed: Mar. 3, 2024. [Online]. Available: http://aakashtechsupportdocs.readthedocs.io/en/latest/nonfunc.html
[181] *'Software Requirements Specification'*. Accessed: Mar. 3, 2024. [Online]. Available: https://belitsoft.com/php-development-services/softwarerequirements-specification-document-example-internationalstandard
[182] S. L. Lim and A. Finkelstein, "StakeRare: Using social networks and collaborative filtering for large-scale requirements elicitation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 3, pp. 707–735, May 2012.
[183] *Semeval-2015 Task 12: Aspect Based Sentiment Analysis*. Accessed: Mar. 3, 2024. [Online]. Available: https://alt.qcri.org/semeval2015/task12/
[184] *Semeval-2016 Task 5: Aspect Based Sentiment Analysis*. Accessed: Mar. 3, 2024. [Online]. Available: https://alt.qcri.org/semeval2016/task5/
[185] *25th IEEE International Requirements Engineering Conference*. Accessed: Mar. 3, 2024. [Online]. Available: http://ctp.di.fct.unl.pt/RE2017/pages/submission/data_papers/
[186] G. Budi Herwanto, G. Quirchmayr, and A. Min Tjoa, "Leveraging NLP techniques for privacy requirements engineering in user stories," *IEEE Access*, vol. 12, pp. 22167–22189, 2024.
[187] F. Dalpiaz. (2018). *Requirements Data Sets (User Stories)*. [Online]. Available: [Online]. Available: http://ctp.di.fct.unl.pt/RE2017/pages/submission/data_papers/
[188] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "PURE: A dataset of public requirements documents," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 502–505.
[189] J. Fischbach, B. Hauptmann, L. Konwitschny, D. Spies, and A. Vogelsang, "Towards causality extraction from requirements," in *Proc. IEEE 28th Int. Requirements Eng. Conf. (RE)*, Aug. 2020, pp. 388–393.
[190] J. Cleland-Huang, M. Vierhauser, and S. Bayley, "Dronology: An incubator for cyber-physical systems research," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng., New Ideas Emerg. Technol. Results (ICSE-NIER)*, May 2018, pp. 109–112.
[191] J. Sayyad Shirabad and T. Menzies, "The PROMISE repository of software engineering databases," School Inf. Technol. Eng., Univ. Ottawa, Ottawa, ON, Canada, Tech. Rep., 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository

VAISHALI SIDDESHWAR (Graduate Student Member, IEEE) received the bachelor's degree in computer science from a premier university in India. She is currently pursuing the Master of Applied Science in electrical engineering with Ontario Tech University. Her thesis focuses on applying artificial intelligence techniques to solve challenges in the requirements engineering domain. She was a Software Engineer with a major organization for many years. Her research interests include requirements engineering, artificial intelligence, deep learning, natural language processing, large language models, and big data.

SANAA ALWIDIAN received the Ph.D. degree in computer science from the University of Ottawa. She is an Assistant Professor with the Department of Electrical, Computer, and Software Engineering, Ontario Tech University. Her work has been published in top peer-reviewed journals and conferences. Her research interests include software engineering, requirements engineering, model-based system engineering, software evolution, AI applications in software engineering, and natural language processing.

MASOUD MAKREHCHI (Member, IEEE) received the B.Sc. degree in electrical engineering from Iran University of Science and Technology, the M.Sc. degree in computer engineering from Shiraz University, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo. He is an Associate Professor with the Department of Electrical, Computer, and Software Engineering, Ontario Tech University. He was a Senior Research Scientist with Thomson Reuters. He has published numerous articles in reputable journals. His research interests include natural language processing, artificial intelligence, machine learning, text and data mining, social computing, mining social networks and complex systems, network science, and moral AI.