Hindawi
Security and Communication Networks
Volume 2020, Article ID 8830683, 19 pages
https://doi.org/10.1155/2020/8830683

Review Article

A Systematic Literature Review on Using Machine Learning Algorithms for Software Requirements Identification on Stack Overflow

Arshad Ahmad,[1,2] **Chong Feng**,[1] **Muzammil Khan**,[3] **Asif Khan**,[1] **Ayaz Ullah**,[2] **Shah Nazir**,[2] **and Adnan Tahir**[4]

[1]*School of Computer Science & Technology, Beijing Institute of Technology, Beijing, China*
[2]*Department of Computer Science, University of Swabi, Anbar, Pakistan*
[3]*Department of Computer Science, University of Swat, Mingora, Pakistan*
[4]*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China*

Correspondence should be addressed to Chong Feng; fengchong@bit.edu.cn

Received 13 May 2020; Revised 1 June 2020; Accepted 25 June 2020; Published 15 July 2020

Academic Editor: Iqtadar Hussain

Context. The improvements made in the last couple of decades in the requirements engineering (RE) processes and methods have witnessed a rapid rise in effectively using diverse machine learning (ML) techniques to resolve several multifaceted RE issues. One such challenging issue is the effective identification and classification of the software requirements on Stack Overflow (SO) for building quality systems. The appropriateness of ML-based techniques to tackle this issue has revealed quite substantial results, much effective than those produced by the usual available natural language processing (NLP) techniques. Nonetheless, a complete, systematic, and detailed comprehension of these ML based techniques is considerably scarce. *Objective*. To identify or recognize and classify the kinds of ML algorithms used for software requirements identification primarily on SO. *Method*. This paper reports a systematic literature review (SLR) collecting empirical evidence published up to May 2020. *Results*. This SLR study found 2,484 published papers related to RE and SO. The data extraction process of the SLR showed that (1) Latent Dirichlet Allocation (LDA) topic modeling is among the widely used ML algorithm in the selected studies and (2) precision and recall are amongst the most commonly utilized evaluation methods for measuring the performance of these ML algorithms. *Conclusion*. Our SLR study revealed that while ML algorithms have phenomenal capabilities of identifying the software requirements on SO, they still are confronted with various open problems/issues that will eventually limit their practical applications and performances. Our SLR study calls for the need of close collaboration venture between the RE and ML communities/researchers to handle the open issues confronted in the development of some real world machine learning-based quality systems.

1. Introduction

The RE activity is steered in the very first phase of software development lifecycle and plays a very pivotal role in ensuring the development of quality and secure software systems [1, 2]. There are various activities (i.e., elicitation, specification, validation, and management) associated with it that need to be effectively performed to somehow guarantee developing a quality software [1–8]. There has been a rapid surge in the RE community of effectively using the diverse online user feedback offered on various social media/online platforms, for instance, the Stack Overflow Q&A site [9], Twitter, bug reporting systems, and mobile app stores (i.e., Google's Play Store and Apple's Play Store) [10] as amongst the latent and rich sources of diverse user requirements [11, 12]. SO Q&A online programming community is commonly used by diverse programmers for learning, problem solving, and sharing knowledge on various issues of software development. For instance, the posts of SO can be about pursuing assistance on programming issues/problems, stating a bug in a software development tool, and requesting or recommending new software

requirements (both functional requirements (FRs) and nonfunctional requirements (NFRs)) omitted in the software development tool, which were totally overlooked previously [13].

Normally software requirements are of two types, namely, FRs and NFRs. The research work on the difference between FRs and NFRs is defined and well known; however, the automatic identification and classification of the software requirements stated in different natural language is still a huge challenge [14–21]. The problem situation becomes more complicated and worst when identifying software requirements from the Q&A on SO (unstructured text) [13, 22, 23]. In addition, some other reasons which make this task problematic are the diversity of stakeholders, difference in the terminologies used, structures of the sentences, and the language used to specify the same kind of requirements [14, 24]. These high levels of inconsistencies and the unstructuredness of the Q&A posts text data on SO make the automatic identification and classification more intricate and challenging [13].

To overcome this complex challenge, several RE researchers have suggested software requirements extraction methods or techniques through using and integrating various ML algorithms applicable in several domains: for instance, effectively eliciting software requirements from the Q&A posts text on SO [22, 23], software requirements elicitation from the documents [12], and automatically classifying user requests in diverse crowd sourced requirements [11]. The results obtained from using these different ML algorithms/techniques for software requirements identification/extraction are significant and more specifically for the SO platform. Nonetheless, a complete, systematic, and detailed comprehension of these emerging ML based techniques for identification or recognition of software requirements on SO is currently unavailable in the existing literature [13]. This SLR study is aimed at filling this gap for both researchers and practitioners' community.

For this SLR study, we have thoroughly followed the systematic literature review (SLR) method as our primary research method [25, 26], with the aim of identifying and classifying the available empirical evidence about the use of emerging ML methods or algorithms for diverse software requirements identification on the SO platform for developing quality systems. The SLR method has been used successfully on diverse topics within the area of requirements engineering [27–30].

The research method will assist in getting a more in-depth understanding of different emerging ML algorithms or methods for identifying/recognizing and comprehending software requirements on the SO, facilitating the following:

Identifying research gaps (research opportunities)

Better decision-making (practitioners) when selecting an ML algorithm

Effectively planning the software requirements identification and avoiding pitfalls

This work presented a detailed SLR work of 12 primary studies related to the emerging ML based approaches or techniques for software requirements recognition or identification on the SO platform. The main research goal of this SLR study is to recognize or identify and categorize or classify the type of machine learning algorithms or techniques used for identifying software requirements on the Stack Overflow platform. Our SLR work is aimed at identifying various types of machine learning algorithms or techniques that have been properly utilized to identify the diverse software requirements on the SO, their working, and evaluation mechanisms. These outcomes will ultimately help and enable us to recognize the main complex issues and challenges that need to be properly tackled to enhance the working capabilities of the different machine learning based techniques. The specific contributions of our SLR study worth mentioning are as follows.

Contribution 1: detecting and categorizing the type of software requirements (RQ1) and ML algorithms or techniques used for identifying diverse software requirements on the SO (RQ2)

Contribution 2: identifying the processes (RQ3) and the performance evaluations of the used ML algorithms (RQ4)

Contribution 3: identifying and collecting basic demographic data, for instance, active researchers (DQ1), types of organizations (DQ2) and countries involved (DQ3), and the most frequently reported publication venues (DQ4)

Contribution 4: collecting diverse evidence giving a centralized source to facilitate the research

The goal of our SLR work is broad enough so we divided it into the set of different research questions (RQs) specified as follows.

RQ1: what types of software requirements identified or reported in the selected studies?

RQ2: what are the types of ML algorithms that have been used for identifying software requirements on SO in the selected studies? Do the ML based approaches outperform the non-ML based approaches? Are there any ML based techniques that considerably outperform the other ML based techniques?

RQ3: what are the types of procedures the reported machine learning algorithms use to identify software requirements on SO?

RQ4: what are the methods utilized to assess the performance of the machine learning algorithms applied in the selected studies? What are the performance outcomes of the reported ML algorithms?

The remainder of the research paper is organized as follows: Section 2 basically describes all the related work. The research methodology followed for this study is explained comprehensively in Section 3. In Section 4, we briefly presented the main results and discussion of the SLR study. Section 5 briefly summarizes the key findings, limitations, and open challenges identified in our SLR study. The different types of the validity threats of the SLR study are

discussed in detail in Section 6. Finally, Section 7 concludes the paper and discusses briefly how the key findings of our SLR study can be further effectively used by the researchers and practitioners in their future research endeavors.

2. Related Work

As per our knowledge till now, there is no single SLR study performed on utilizing machine learning algorithms or techniques for identifying software requirements on the SO online Q&A website. There are some surveys available on the SO [13, 31], but as of now none of them are related in any way to ML algorithms or methods for detecting/identifying software requirements on the SO online Q&A website.

In 2018, Ahmad et al. [13] performed an ad hoc literature review, by selecting 166 research papers on the SO that were mainly classified about software development life cycle from the start of the SO website till the year 2016 positively. They categorized the studies on SO platform into two: "SO design and usage" and "SO content applications." This categorization not only gives detailed comprehensions to the SO Q&A forum or platform providers about the possible limitations in the design and usage of such platforms but also offers ways to address them in the days to come. It also empowers software programmers to utilize such platforms for the recognized underutilized different tasks of software development lifecycle, e.g., software requirements specification, design, analysis, and validation.

Similarly, the work of Baltadzhieva and Chrupała [31] thoroughly reviewed and analyzed various questions quality posted on diverse community question answering (CQA) websites like SO. Besides, they pointed out the different metrics through which the quality of the posted questions can be identified, which in large ultimately lead to affecting the question quality.

There are various other surveys available [32–34] that are performed for identifying ML algorithms used for recognizing, collecting, and categorizing/classifying the nonfunctional requirements in software documents. In 2013, Meth et al. [32] conducted an SLR on investigating the works on automated requirements elicitation. They selected 36 primary papers published between January 1992 and March 2012. They categorized the identified works through an analysis framework, including tool categorization, technological concepts, and assessment approaches.

Later on, Binkhonain and Zaho [33] conducted an SLR on ML algorithms for identifying and classifying NFRs. They have selected 24 primary studies published. The key findings of their work revealed that ML approaches could identify and classify NFRs, but they still have many challenges that need more attention. Recently, Iqbal et al. [34] presented a survey on ML algorithms and requirements engineering. They provided a bird's-eye view of how ML algorithms are aiding different requirements engineering activities. The results revealed that the impact of ML algorithms could be found in different phases of RE lifecycle, e.g., software requirements elicitation, analysis, specification, validation, and management.

Besides, there are some surveys, SLR's, and systematic mapping studies done in other areas on sentiment analysis of scientific citations [35], data preprocessing methods for class imbalance problem [36], ML algorithms or techniques based software development effort estimation models [37], usability in agile software development [38, 39], and requirements prioritization [40], among others.

Unlike the different related works cited above, our study is focused on identifying and classifying ML algorithms or techniques used for identifying diverse software requirements on the SO only. We have selected 12 primary studies for our SLR work published until May 2020. Thus, we are the first to perform a comprehensive SLR study aimed at identifying, reviewing, summarizing, assessing, and thoroughly reporting the diverse works of ML algorithms or techniques for identifying diverse software requirements on the SO.

3. SLR Research Methodology

There has been a rapid surge in the popularity of using the Evidence-Based Software Engineering (EBSE) among researchers due to the applicability of systematic literature review (SLR) in various domains [41–43]. The key goal of the SLR study is to systematically identify, classify, and thoroughly synthesize any new evidence based on the data extracted from the selected research publications.

We conducted a comprehensive SLR study, thoroughly following the systematic guidelines defined and stated in [25, 26, 44–46], with the aim of identifying and classifying the types of machine learning algorithms or techniques used for identifying the software requirements on the Stack Overflow. The steps of the SLR research method encompassed the following activities:

(1) Planning for the SLR study (the SLR protocol):
   (a) Proper definition of the research goals and the set of different research questions
   (b) Description of the strategies for search, selection, and data extraction processes
   (c) Identification and description of potential validity threats
   (d) Tasks allocation and assignment (roles and responsibilities of every researcher involved)

(2) Conducting the SLR study (executing the SLR protocol):
   (a) Thoroughly searching for the primary studies
   (b) Selecting cautiously the relevant primary studies
   (c) Data extraction and thorough analysis of the selected primary studies to produce a classification schema (the map)

(3) Reporting the SLR results:

The initial findings of our SLR study have been published previously in a conference [47]; since then, the research questions have been added (demographic research questions) and modified a little bit (specifically RQ2 and RQ4). In addition, we

have also added snowballing search strategy [48] as a complementary strategy in addition to the automated electronic data sources with the aim of not overlooking any relevant paper. Basically, the conference paper just reported the initial findings with no detailed analysis of the results. In this work we have deeply assessed all the findings of the research questions, reported the key findings and limitations, and discussed the open challenges for future researchers. The subsequent subsections give comprehensive information regarding the main activities of the SLR protocol used.

3.1. Goal and Research Questions. The key research goal of our SLR study is to recognize or identify and classify or categorize the different types of machine learning algorithms or techniques used for identifying the software requirements on the Stack Overflow.

The goal of our SLR work is broad enough, so we divided it into the set of different research questions (RQs) specified in Table 1.

We also collected evidence to answer some interesting demographic questions (DQs) as suggested in [44–46] associated with the identification of the most actively participating researchers, organizations affiliations (academia or industry), and countries, as well as the top publication venues. Table 2 presents briefly the description of every DQ.

3.2. Search Strategy (EDS and Snowballing Method). The EBSE technique is totally dependent on the approach of identifying, collecting, and summarizing all the existing empirical evidence. Nonetheless, it is quite hard to fully ensure that all the existing empirical evidence was recognized; we ultimately need to abate the validity threat of not relying on single search strategy [49]. Therefore, two diverse search approaches, i.e., EDS and snowballing, were considered, with the aim of complementing each other, retrieving the highest number (recall) of potentially relevant studies (precision). The relevant experts on the RE and Stack Overflow areas validated the diverse search strategies.

The search was thoroughly performed in four diverse electronic data sources (EDS), namely, the ACM Digital Library, IEEE Xplore, Scopus, and Web of Science (WoS), respectively [50, 51]. These different EDS ensure including the diverse main venues (i.e., conference proceedings, workshops, and journals) specifically in the field of Software Engineering [52, 53]. The automated search strings were developed from the combination of the key terms extracted from our defined research questions, keywords from the different research publications retrieved by a pilot search, and the list of synonyms.

The initial set of search terms were integrated and adopted to the particular EDS with the help of using the "AND" (if allowed) and "OR" (if allowed) logical operators wherever possible. We conducted several search rounds in the diverse EDS until we accomplished the best balance between precision and recall measures. Table 3 presents the set of final selected search strings, adapted to each of the four electronic data sources, respectively. The automated search strategy was carried out on 23/08/2019 and successfully retrieved 1,073 results only.

We also performed both backward and forward snowballing search methodology as outlined in [48, 54] in addition to the automated electronic databases search strategy on 27/05/2020. To start the snowballing process, the 12 primary studies were used as initial seeds that were selected from the automated search strategy. In the snowballing process, it is vital to consider only the suitable or relevant research studies, so to ensure this, we adopted a top-down sequential process to include only the relevant research papers for each stage of the new iteration. To guarantee the relevance, we also performed the data extraction with the aim that the preselected papers were suitable for answering the defined research questions. All those papers were selected as the new seeds for the next stage or iteration of the snowballing process which passed the data extraction criterion.

Finally, the snowballing process retrieved 1,411 papers and ended the process at the third iteration with no new primary papers found. To minimize the possible biasness, two of the authors individually conducted the selection of the papers, and a third author carefully reviewed the data generated from every iteration, integrated the individual outcomes, and thoroughly assessed them for any disagreements. Tables 4 and 5, respectively, present the outputs of the two adopted search approaches for the SLR study.

3.3. Research Paper Selection Criteria (Inclusion and Exclusion Criteria). The selection process primarily consists of two tasks: a principally perfect definition of both inclusion and exclusion criteria and truly applying these definite benchmarks to select the pertinent primary research studies [55, 56]. As inclusion and exclusion are principally two conflicting activities, we chose to categorically focus our efforts on the exclusion criterion, by outlining a clear set of criteria, both objectively and subjectively appropriate. The former one does not cause any sort of threat to the validity, and, henceforth, its application is much easier and simpler. While applying the very first exclusion criterion, specifically, those related to the language and duplicity assisted us to remove irrelevant data quite rapidly. For this SLR study, the following objective exclusion criteria were applied to all the retrieved papers:

(a) Exclusion criterion 1: research papers not written in English language

(b) Exclusion criterion 2: short research papers (less than four pages in length)

(c) Exclusion criterion 3: research papers not published in peer-reviewed publication venues

(d) Exclusion criterion 4: research papers that are not a primary research study (secondary and tertiary research studies)

Table 1: Description of the research questions.

Research question	Description
RQ1: what are the types of software requirements that are identified or reported in the selected studies?	The different types of software requirements are functional requirements (FRs) and nonfunctional requirements (NFRs), or others.
RQ2: what are the types of ML algorithms that have been used for identifying software requirements on SO in the selected studies? Do the ML based approaches outperform the non-ML based approaches? Are there any ML based techniques that considerably outperform the other ML based techniques?	To identify all the techniques or methods (algorithms) used in the selected primary studies.
RQ3: what are the types of procedures the reported machine learning algorithms use to identify software requirements on SO?	To know all the processes (natural language processing); those are used in the selected primary studies.
RQ4: what are the methods utilized to assess the performance of the machine learning algorithms applied in the selected studies? What are the performance outcomes of the reported ML algorithms?	To know the different performance evaluation criteria applied in the selected primary studies, their results, strengths, and weaknesses, respectively.

Table 2: Descriptions of the demographic questions.

Demographic questions	Description
DQ1: who are the most actively participating researchers?	All authors, ordered by the number of papers
DQ2: which organizations are the most active?	Based on the affiliations of all the authors
DQ3: which countries are the most active based on authors affiliations?	Based on the affiliations of all the authors
DQ4: which are the top venues for the publications?	Type (conference, journal, or workshop) and the name of the publishing venue

Table 3: Set of search strings, adapted to each of the EDS.

EDS	Search strings
IEEE Xplore	(("Document Title" : "requirements*") OR ("document Title" : "NFR") OR ("document Title" : "functional requirements") OR ("document title" : "non-functional requirements") OR ("document title" : "quality requirements")) AND (("Abstract" : Stackoverflow) OR ("Abstract" : "stack overflow") OR ("Abstract" : techniques) OR ("Abstract" : "machine learning")) AND (("author Keywords" : "requirements*") OR ("author Keywords" : "stack overflow") OR ("author Keywords" : stackoverflow) OR ("author Keywords" : "NFR") OR ("author keywords" : "non-functional") OR ("author keywords" : "functional"))
ACM Digital Library	(acmdlTitle : (+requirements* + NFR + quality + functional + non-functional) OR acmdlTitle : (Stackoverflow stack overflow)) and recordAbstract : (algorithm techniques "machine learning") OR keywords.author.keyword : (+requirements* + stackoverflow + functional + non-functional + FR + NFR))
Web of Science (WoS)	TI = ("requirements engineering" OR "NFR" OR "non-functional requirements" OR "quality requirements" OR "functional requirements") AND TS = (Stack overflow OR Stackoverflow OR "machine learning" OR algorithms OR techniques) and language: (english) refined by: web of science categories: (computer science software engineering)
Scopus	TITLE ("requirements engineering" OR NFR OR non-functional OR functional OR quality) and TITLE-ABS-KEY (stack overflow OR stackoverflow OR "machine learning" OR techniques OR algorithms) AND (LIMIT-TO (SUBJAREA, "COMP")) AND (LIMIT-TO (LANGUAGE, "English"))

(e) Exclusion criterion 5: any kind of grey literature (books, presentations, poster sessions, forewords, talks, editorials, tutorials, panels, keynotes, etc.)

(f) Exclusion criterion 6: all sorts of research thesis whether Ph.D. or master or bachelor theses

It is obvious that subjective criteria are very complex to address adequately in any SLR study including this one. They are prone to create biasness into the SLR study, and, thus, a predefined protocol principally needs to be applied with the aim of minimizing this threat. On the contrary, applying these criteria might also leads to a substantial reduction in the number of research papers to consider as being relevant. For this SLR study, the authors applied the two exclusion criteria described as follows:

(a) Not focus: research studies not related to any of the RE activities on the Stack Overflow

(b) Out of scope: research studies not related to any of the RE phases of software development lifecycle

Any research paper not excluded by the aforementioned criteria was deemed relevant and included in the set of final selected primary research studies. The authors primarily adopted a top-down method to the application of these criteria on the research papers. In the first stage, some metadata information such as the title, abstract, and keywords of the research paper was taken into consideration. If these data were not sufficient to exclude any research paper at hand, then the authors reviewed the full text of the research publication, more specifically the introduction

Table 4: Automated searching in EDS results.

Source	Works retrieved
ACM Digital Library	269
IEEE Xplore	481
Scopus	104
Web of Science	219
Total	1,073

Table 5: Snowballing method search results.

Snow balling iterations	Number of seeds	Number of citations
First iteration	12	783
Second iteration	82	624
Third iteration	3	4
Total	97	1,411

(problems and contributions of the research study), the results, and conclusions sections of the research study.

To handle appropriately with any disagreements, the authors primarily followed the inclusive criteria as systematically suggested in [44] and described in detail in Table 6. The authors excluded a research publication at hand only when both of the reviewers agreed (category "F") or marked the research publication as the borderline (category "E").

The complete diagrammatic flow of both the searches performed (EDS and Snowballing), detailed systematic selection processes and the outcome of every task of the SLR study are reflected in Figure 1. A final set of 12 research papers was selected for this SLR study (the detailed list with full bibliographic references is provided in Table 7). Besides, the details of the quality assessment criterion adopted for the SLR study are described in the next section.

3.4. Quality Assessment Criteria. For any research publication to pass the defined selection phase, a comprehensive quality assessment criterion was defined. The authors defined principally "4" quality assessment criteria questions to assess the rigorousness, reliability, and significance of the relevant studies as suggested by [6, 57–59] for the research paper as shown in Table 8. Thus, a research study which accomplished a quality score of "4" was thus considered in the final selection.

3.5. Data Extraction Process. To minimize any sort of biasness in the data extraction process, one of the authors' developed a comprehensive data extraction form (DEF) in the spread sheet format. The DEF (see Table 9) was mainly used to extract and store the data for each of the selected research studies. The rest of the authors thoroughly reviewed, improved, and agreed upon the DEF before properly starting the data extraction process. The proper use of the DEF facilitates a detailed, systematic, explicit, and consistent approach to conduct the data extraction process of an SLR study [60, 61].

Table 6: Dealing with disagreements.

		Reviewer X		
		Include	Uncertain	Exclude
Reviewer Y	Include	A	B	D
	Uncertain	B	C	E
	Exclude	D	E	F

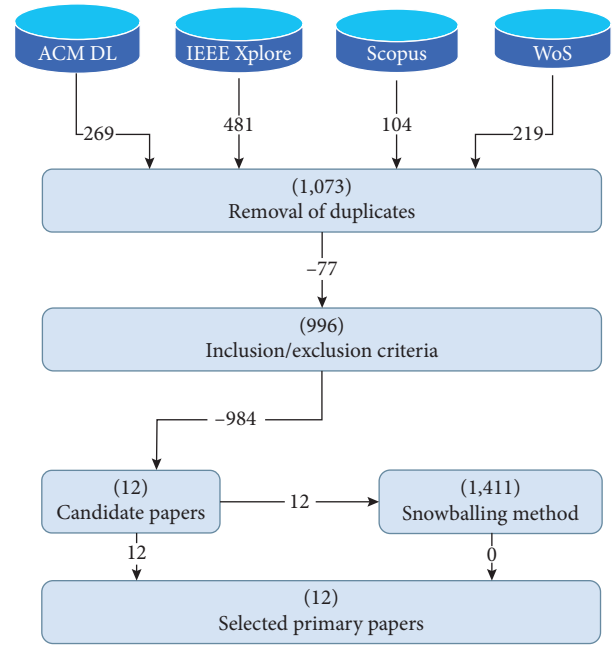

Figure 1: Summary of the searches and selection processes of the SLR study.

To conduct the data extraction process properly, the authors distributed the set of selected primary research publications into two halves, as stated below.

The first half of the research papers was blindly allocated to the reviewers R1 (the first author) and R2 (the second author), respectively. Both of the reviewers R1 and R2 thoroughly gauged the work independently and, after finishing their respective tasks, fixed all the discrepancies (if any) to yield an agreed dataset. The second half of the research papers, too, was blindly allocated to the reviewers R1 (the first author) and R3 (third author), respectively. Each pair of the reviewers populated the DEF individually. When any sort of disagreements arose, a consensus meeting was arranged, with the participation of a third reviewer, until all (if any) discrepancies were fixed [62].

4. Results and Discussion

In this section, the authors present and discuss briefly the results of the SLR study. For each of the RQs, the authors instigate them with a summary of the most noteworthy results, a discussion about the most relevant facets, and, based on them, a suggestion of some explanatory hypotheses.

Table 7: List of selected primary studies.

Paper ID	Full bibliographic reference
S01	Yin, H., and Pfahl, D. (2017, November). A preliminary study on the suitability of stack overflow for open innovation in requirements engineering. In Proceedings of the 3rd International Conference on Communication and Information Processing (pp. 45–49). ACM.
S02	Zou, J., Xu, L., Guo, W., Yan, M., Yang, D., and Zhang, X. (2015, May). Which non-functional requirements do developers focus on? An empirical study on stack overflow using topic analysis. In 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories (pp. 446–449). IEEE.
S03	Pinto, G., castor, F., and liu, Y. D. (2014, May). Mining questions about software energy consumption. In Proceedings of the 11th Working Conference on Mining Software Repositories (pp. 22–31). ACM.
S04	Ahmad, A., li, K., Feng, C., and sun, T. (2018, October) "An empirical study on how iOS developers report quality Aspects on stack overflow," international journal of machine learning and computing vol. 8, no. 5, pp. 501–506, 2018. IJMLC.
S05	Treude, C., Barzilay, O., and storey, M. A. (2011, May). How do programmers ask and answer questions on the web?: Nier track. In 2011 33rd International Conference on Software Engineering (ICSE) (pp. 804–807). IEEE.
S06	Zou, J., Xu, L., Yang, M., Zhang, X., and Yang, D. (2017). Towards comprehending the non-functional requirements through developers' eyes: An exploration of stack overflow using topic analysis. Information and Software Technology, 84, 19–32.
S07	Bajaj, K., pattabiraman, K., and mesbah, A. (2014, May). Mining questions asked by web developers. In Proceedings of the 11th Working Conference on Mining Software Repositories (pp. 112–121). ACM.
S08	Ahmad, A., Feng, C., li, K., Asim, S. M., and sun, T. (2019). Toward empirically investigating non-Functional requirements of iOS developers on stack overflow. IEEE Access, 7, 61145–61169.
S09	Xiao, M., Yin, G., wang, T., Yang, C., and chen, M. (2015). Requirement acquisition from social q&a sites. In Requirements Engineering in the Big Data Era (pp. 64–74). Springer, Berlin, Heidelberg.
S10	Rosen, C., and shihab, E. (2016). What are mobile developers asking about? A large scale study using stack overflow. Empirical Software Engineering, 21(3), 1192–1223.
S11	Abad, Z. S. H., shymka, A., pant, S., currie, A., and ruhe, G. (2016, September). What are practitioners asking about requirements engineering? An exploratory analysis of social q&a sites. In 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW) (pp. 334–343). IEEE.
S12	Pinto, G. H., and Kamei, F. (2013, October). What programmers say about refactoring tools?: An empirical investigation of stack overflow. In Proceedings of the 2013 ACM workshop on Workshop on refactoring tools (pp. 33–36). ACM.

Table 8: Quality assessments.

Quality criteria	Score
Are the findings and results explicitly mentioned in the study?	Yes = 1 No = 0
Is there any empirical evidence on the findings in the study?	Yes = 1 No = 0
Are the arguments well-presented and justified in the study?	Yes = 1 No = 0
Is the study well referenced?	Yes = 1 No = 0

4.1. RQ1: What Are the Types of Software Requirements Identified or Reported in the Selected Studies? For the data extraction process, the authors started with a preestablished classification scheme of software requirements to group the types of reported software requirements. The classification scheme includes primarily two types of software requirements: that is, functional requirements (FRs) and nonfunctional requirements (NFRs). The authors found direct evidence about the presence of both types of software requirements, FRs and NFRs, on the SO. The authors identified seven primary papers (S01, S03, S07, S09, S10, S11, and S12) related to functional requirements and five studies (S02, S04, S05, S06, and S08) related to nonfunctional requirements identified on the SO, as depicted in Table 10. Table 7 shows the full list of selected papers.

The majority (approx. 58.33 %) of the selected primary studies specifically focused on extracting functional requirements (FRs) related to general software development tools. Some examples of the identified and extracted FRs questions were about extracting developers' needs of Eclipse IDE tools (e.g., S01), suggesting features (e.g., S03), reporting issues (e.g., S07), and requesting new features (e.g., S09, S10, S11, and S12) among others. The remaining approximately 41.66% of the selected primary studies, identified the nonfunctional requirements (NFRs) on the SO. Some examples of the identified/recognized NFRs were primarily related to the mobile application development tools (e.g., S02, S04, S06, and S08) and the web (e.g., S05).

4.2. RQ2: What Are the Types of ML Algorithms That Have Been Used for Identifying Software Requirements on SO in the Selected Studies? Do the ML-Based Approaches Outperform the Non-ML-Based Approaches? Are There Any ML-Based Techniques That Considerably Outperform the Other ML-Based Techniques? There is a plethora of diverse machine learning algorithms and techniques available for text processing; however, they can be classified primarily into two main categories named as the supervised learning (SL) and the unsupervised learning (USL) algorithms/techniques. The aim of the supervised learning algorithm is to deduce a function from the labelled training comprised of data instances and the anticipated outcome value for each of the instances. Then, the inferred function could be utilized to predict the label for all the hidden data instances [63]. On the contrary, the unsupervised learning algorithm deduces a function to specify the unseen structure of the unlabelled data [63]. Hence, it explores the data for similar patterns,

Table 9: Description of the data extraction form.

Data item	Description
Bibliographic information	Authors, title of the paper, name and type of the organization, country, publication type and venue, and year of publication.
Study goal	The main intent of the selected primary research studies.
ML algorithm/technique	The different types of the ML algorithms/techniques employed to identify and classify the software requirements.
Types of the requirements	The different types of software requirements FRs/NFRs.
Process/approach	The process/approach used to identify the FRs/NFRs from posts.
Performance evaluation	How were the performance and efficiency of the ML algorithms measured in the selected primary research studies and what were their actual outcomes?

Table 10: Papers reporting different types of software requirements.

Primary papers	Type of software requirements	Support level
S01, S03, S07, S09, S10, S11, S12	FRs	7
S02, S04, S05, S06, S08	NFRs	5

and then grouping (clustering) of the similar instances is produced. Besides, there is another type called the semi-supervised learning (SSL) that lies between the supervised and the unsupervised learning algorithms/techniques. The aim in the semisupervised learning is to deduce a function grounded on lesser amount of labelled training dataset and huge amount of unlabelled dataset [63].

The Latent Dirichlet Allocation (LDA) is one of the most widely adopted and naïve statistical topic modeling algorithms. The basic idea behind LDA is that each document in the text corpus may be related to several different topics. LDA is clearly defined by its generative process, the process by which it is expected that each text document of the corpus is produced. In this generative model, it is supposed that each text document of the corpus is created mainly in two stages: in the first stage, the set of topics are selected for the text document based on the distribution of the topics. In the second stage, to generate each word in the text document, a topic is selected based on the distribution in the first stage, and then a word is arbitrarily chosen conferring to the distribution of words for each of the topics. In this manner, it generates a set of words and places it together as a text document. In this generative model, each of the text documents is characterized as a bag-of-words, that is, a naive characterization of a text as a multiset of words, ignoring grammar and the ordering of the words in the text document [64, 65].

The Support Vector Machines (SVM) are a group of SL ML algorithms, primarily utilized for classifications or/and regressions. The SVM method is basically grounded upon the statistical learning theory and the certain dimensions mentioned in [66, 67]. The purpose of the SVM technique is to create the optimal separating line aimed at categorizing all of the input data into various classes.

The thematic analysis is not an ML algorithm but a process/technique (manual or automatic) that involves exploring, identifying, extracting, and storing the patterns (or themes) that emerge within data at hand. Besides, the identified themes are basically the patterns that emerge across the datasets significant to the description of an occurrence and are related to a specific research question (RQ). Nevertheless, the themes are recognized by combining the modules or pieces of concepts or experiences, which are usually worthless when examined in separation. Finally, these themes ultimately serve as the categories for the analysis [68–70].

The outcome of this RQ revealed that there were mainly two different machine learning algorithms identified in the selected primary research studies. These machine learning algorithms basically fall into two types, comprising one SL, and eight USL ML algorithms. Besides, we also found three primary studies that used thematic analysis or qualitative coding techniques, as comprehensively summarized in Table 11. Table 11 also depicts that USL algorithms, specifically LDA, are the most popular/common type of machine learning algorithm, as they were utilized in eight primary research studies (averaging approximately 67%).

Besides, the FRs were identified in the four studies (e.g., S01, S07, S10, and S11) and the NFRs were also identified in the four selected studies (e.g., S02, S04, S06, and S08), respectively, out of the eight primary studies that used the LDA algorithm. SVM was the second most popular ML algorithm category (approximately 8.3%). It was quite interesting to observe that three of the selected primary studies (25%) did not use any algorithm and have used thematic coding or qualitative coding technique for identifying and classifying the different types of software requirements on the SO.

The detailed performance evaluations of all the ML algorithms and techniques of the selected studies of our SLR study are discussed in the results and discussion section of RQ4 (see Section 4.4).

4.3. RQ3: What Are the Types of Procedures the Reported Machine Learning Algorithms Use to Identify Software Requirements on SO? The outcomes of the detailed analysis of the 12 selected primary research studies of the SLR study have depicted a general process pattern for utilizing or

Table 11: Different ML algorithms.

Type	ML algorithms/ technique	Purpose	Type of software requirements	Support level
USL	LDA	Finding the hidden or latent topic(s) that the documents (posts) contain based on their probabilities	FRs (S01, S07, S10, and S11), NFRs (S02, S04, S06, and S08)	8
	Thematic coding	It involves exploring, identifying, extracting, and storing the patterns or themes	FRs (S03 and S12), NFRs (S05)	3
SL	SVM	It is primarily used for classification and is grounded upon the statistical learning theory	FRs (S09)	1

applying the machine learning algorithms approach to recognize or identify and categorize or classify the different types of software requirements on the SO. This whole process could possibly be classified into three (3) main stages or phases: the text preprocessing phase is the phase in which the irrelevant text from the data is removed (RQ3); the learning phase basically involves applying the actual different ML algorithms (addressed in RQ2); and finally the evaluation phase (addressed in RQ4) involves assessing or evaluating an ML algorithm's approach to identify or recognize and categorize or classify the different types of software requirements on the SO as depicted in Figure 2.

The text preparation phase is the text preprocessing phase, where it takes textual data of SO Q&A as input and applies some NLP techniques to clean the textual data for further processing. A total of six different NLP preprocessing techniques were identified in the selected research papers as depicted in Table 12. The identified different preprocessing techniques are explained briefly as follows:

Stop words removal: it is the process of removing certain auxiliary verbs (e.g., be, do, and have), conjunctions (e.g., and, or), and different articles (e.g., the, a, and an) from the text [71]

Case unification: it is the process of converting the text into a uniform style, i.e., lowercase or uppercase

Tokenization: it is the process of splitting the sentence into words [71]

Stemming: it is the process of stemming or reducing the words to their origin or roots. For instance, words like 'goes,'' gone,' and "going" will be reduced to 'go' [72]

Punctuations removal: it is the process of removing different punctuations (e.g., commas, semicolons, question marks, and exclamation marks) from the text

The outcome of this research question (RQ) revealed that approximately 66.67% of the selected studies used different preprocessing techniques. The remaining studies averaging approximately (33.33%) lack reporting about using any of the preprocessing steps. The different preprocessing approaches or techniques used in the selected research studies are the stop words removal (approx. 58.33%), case unification (approx. 50%), tokenization (approx. 41.66%), stemming (approx. 25%), punctuations removal (approx. 16.67%), and filtration (approx. 8.33%) as depicted in Table 12. These findings reveal that there is lack of using different appropriate and uniform NLP techniques for preprocessing the SO Q&A text.

As depicted in Figure 2, predefined data is a prerequisite for building or constructing FRs/NFRs classifiers using the various ML algorithms. Specifically, the SL machine learning algorithms need a prelabelled dataset, whereas the USL machine learning algorithms need a set of predefined groups or keywords. Besides, the SSL machine learning approaches, alike the SL machine learning approaches, also need a prelabelled dataset. This process is not been explicitly stated in the surveyed techniques/approaches of this SLR research study. In addition, most of the selected primary research studies treat the machine learning based methods as 'black boxes' and offer no explicit explanation on how actually all these machine learning methods work.

In the evaluation phase, the assessment of both the FRs/NFRs classifier is performed using several methods to find the efficacy of the classifiers. Besides, the performance evaluation of the techniques and results is discussed in the results and discussion section of RQ4 (see Section 4.4).

4.4. RQ4: What Are the Methods Utilized to Assess the Performance of the Machine Learning Algorithms Applied in the Selected Studies? What Are the Performance Outcomes of the Reported ML Algorithms? Out of the 12 selected research studies of this SLR study, merely 6 of them (averaging approx. 50%) have stated their performance evaluation criterion. Table 13 briefly sums up the four performance evaluation criteria recognized from the selected research studies and their respective outcomes.

It was quite significant to observe (as depicted in Table 13) that though the LDA and SVM were used in various selected research papers, their performance results considerably vary from one another. For instance, LDA performed well in S08; however it did not perform that good in the S11. In the S04, LDA got a fairly well recall outcome compared to the precision outcome. Nevertheless, all of the selected studies achieved considerably good recall results compared to the precision results irrespective of the ML algorithm applied, that is, LDA and SVM.

Amongst the four different performance evaluation criteria, the precision and the recall are deemed amongst the famous ones, trailed by the F-measure and the word and topic intrusion. In the following, we briefly describe how effectively the four measures are used to enumerate and

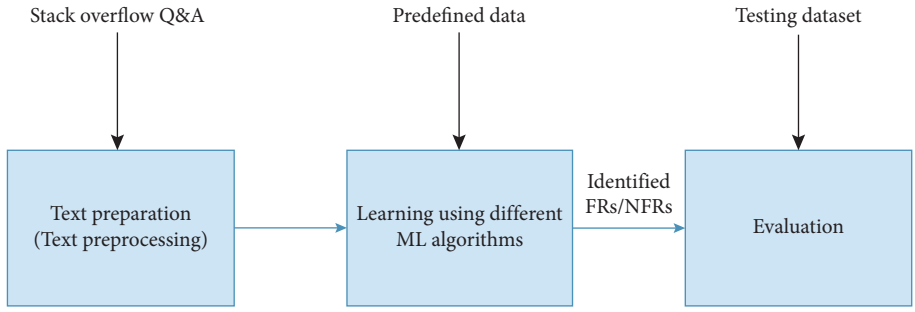

Figure 2: A generic process of using MLl algorithms to identify software requirements on the SO.

Table 12: NLP techniques used for text processing.

NLP process	Paper ID	Support level
Stop words removal	S02, S04, S06, S07, S08, S09, and S11	7
Case unification	S02, S04, S06, S08, S09, and S11	6
Tokenization	S02, S04, S06, S08, and S09	5
Stemming	S07, S09, and S11	3
Punctuations removal	S09 and S11	2
Filtration	S10	1

Table 13: Performance evaluations used.

ML algorithm	Study ID	Precision (%)	Recall (%)	F-measure (%)	Word and topic intrusion (%)
LDA	S02	68.7	75.9		
	S04	70.33	77		
	S06	68.7	75.9		
	S08	70.33	77		
	S11				58
SVM	S09	72	77	74.42	

evaluate the performances of the described ML techniques/algorithms in the 12 selected research studies of the SLR study.

4.4.1. Precision and Recall Measures. Out of the 12 selected research papers of this SLR study that state their diverse performance measures, merely 4 of them (averaging approx. 41.66%) have properly employed these two matrices with the aim of effectively assessing the performance of the utilized machine learning techniques and algorithms. The precision measure could be defined as the aggregate number of correctly classified FRs/NFRs in ratio to the number of FRs/NFRs retrieved. It is mathematically represented as P = true positive/(true positive + false positive). Recall measures can be defined as the ratio of FRs/NFRs that have been correctly categorized/distributed and mathematically can be represented as R = true positives/(true positives + false negatives) [73]. True positive basically characterizes the number of correctly classified software requirements (FRs/NFRs), the false positive characterizes the amount of incorrectly categorized/distributed diverse software requirements (FRs/NFRs), and true negative signifies the amount of diverse software requirements (FRs/NFRs) correctly not categorized/distributed, whereas false negative is basically the amount of diverse software requirements (FRs/NFRs) incorrectly not classified [74].

The precision and recall evaluation measures are mostly used together [75], and there always exists a trade-off between them. Besides, the goal of the precision evaluation measure is to ensure that all of the retrieved diverse software requirements are correctly relevant, while the recall aims at retrieving all of the relevant diverse software requirements. Nevertheless, which value (Precision or Recall) is highly significant to quantify the identification or classification results can be considerably argued. The research studies S02, S04, S06, S08, and S09 emphasized on accomplishing a high recall value as it was considered that the accuracy of automatic software requirements labeling (FRs/NFRs) in the topics/posts is important. It is worth mentioning that the selected studies S02, S04, S06, S08, and S09 have achieved high recall rate which indicates the acceptability of accurately labeling or classifying the requirements (FRs/NFRs) identified. Among the selected 12 studies, S09 achieved high recall and precision rates followed by S04 and S08, which signifies the performance of the ML algorithms used.

Nevertheless, S09 viewed that precision was also equally significant if the recall was satisfactory for the automatic categorization/classification to ignore huge number of irrelevant diverse software requirements (FRs/NFRs) from

being wrongly classified as pertinent (false positives). It is also evident from the values of both the precision and the recall as depicted in Table 13 that there are no studies that achieved higher recall and considerable lower precision results, or there were no such research papers that have achieved considerably higher precision and considerably lower recall results.

4.4.2. F-Measures. The F-measure takes into consideration both the precision and the recall measures and is basically the weighted average of the precision and recall measures. F-measure can be defined as F-measure = 2 × (precision × recall)/(precision + recall). We can also define the F-measure as the harmonic mean of precision and recall measures [76]. Only one (8.33%) of the selected primary research studies applied the F-measure. However, they did not give explicitly the reasons and importance of employing this measure in their study. They only claimed that this measure is commonly used for information extraction tasks. Nevertheless, they utilized this measure to integrate the precision and recall measures as the F-measure corresponds to the weighted average value of precision and recall measures [77] and then used the output value as a gauge for the performance evaluation.

4.4.3. Word and Topic Intrusion. Only one (averaging approx. 8.33%) of the selected studies used the word intrusion and topic intrusion method for evaluating the performance of the applied ML algorithm. The word intrusion method basically assesses the quality of the inferred topic(s) by computing their level of "cohesiveness." The topic intrusion method assesses whether the content of a document matches with the topic(s) it has been allotted to, conferring to the human judgment [78]. The authors did not give any reasons for employing this measure in their study and just claimed to assess the performance of the used ML algorithm (topic modeling) following the stepwise guidelines of [78].

4.5. DQs: Who Are the Most Actively Participating Researchers? A total of 45 researchers appear as authors in the 12 primary papers selected by our study. We only report the top 10 active researchers, based on their number of published works. It is worth stating that the data allowed us to identify groups of researchers who often published collaborative works (e.g., Zou, Xu, Zhang and Yang and Ahmad, Li, Feng and Sun). Besides, all of the active researchers are affiliated with academic organizations. It is also worth mentioning that all of the active researchers, those with two publications, are based in China except one author (Pinto) who is based in Brazil. Moreover, all of the identified researchers are currently active. The identified top ten active researchers and their affiliations are depicted in Table 14.

4.6. DQs: Which Organizations Are the Most Active? We have identified a total of 13 different organizations based on the number of mentions in the selected studies. All of the identified organizations are academic institutions, and the majority is based in China. Table 15 shows a list of the ten most active organizations. It is noteworthy that there is no single study reported from the industry; thus, there is a huge imbalance and gap between the amount of research from academia and industry (100% academic, 0% industry), which highlights the need for research in industrial contexts, or in academia-industry collaborations.

4.7. DQs: Which Countries Are the Most Active Based on the Authors' Affiliations? Based on the affiliations of all the authors, we have been able to identify seven different countries located in the four continents. The top three countries with the greatest representation are China, Canada, and Brazil. The rest of the countries do not reach a representation higher than 50% of the previous three. Figure 3 shows the number of authors whose affiliations belong to the seven countries with the number of mentions.

4.8. DQs: Which Are the Top Venues for the Publications? The outcome of this DQ revealed that the favorite venues for publication of the selected studies were mainly conferences with 50% of the primary works as depicted in Figure 4. The ACM/IEEE Working Conference on Mining Software Repositories, with the most significant primary papers published, topped the list of the most cited, as shown in Table 16. Combining with the workshops (16.7%), the conferences cover approximately 66.7% of the selected studies. Nonetheless, 33.3% of the papers, published in journals, are a worthy fact and provide a solid empirical basis, considering the quality and the standard prestige of publication venues (e.g., Information and Software Technology, Empirical Software Engineering, and IEEE Access). Table 16 shows the complete list of mentioned conferences, journals, and workshops.

Moreover, Figure 5 shows that the interest in identifying software requirements on SO has remained at a moderate level (approximately two papers per year) since 2014, with higher peaks in 2014, 2015, and 2017, respectively.

5. Findings, Limitations, and Open Issues

In this section, we will primarily discuss in brief some of the key research findings, the limitations of the reviewed ML approaches, and finally the open issues from our SLR results.

5.1. Key Findings of the SLR. Some of the significant findings that can be drawn from the SLR study are mentioned as follows:

The identified ML based techniques/approaches have considerably performed well by accomplishing an accuracy of approximately more than 70% in detecting/identifying and categorizing/classifying the software requirements (FRs and NFRs) on SO.

Overall, the SVM (SL algorithm) performed better than LDA (USL algorithm) though having the same recall

Table 14: Active researchers based on number of papers published.

Author's name	Organization	Support level
Pinto	Federal University of Pernambuco, Brazil	2
Zou	The School of Software Engineering, Chongqing University, China	2
Zhang	The School of Software Engineering, Chongqing University, China	2
Yang	The School of Software Engineering, Chongqing University, China	2
Xu	The School of Software Engineering, Chongqing University, China	2
Ahmad	Beijing Institute of Technology, China	2
Feng	Beijing Institute of Technology, China	2
Li	Beijing Institute of Technology, China	2
Sun	Beijing Institute of Technology, China	2
Yin	Institute of Computer Science, University of Tartu, Estonia	1

Table 15: Top 10 active organizations.

Organization's name and country	No. of mentions
The School of Software Engineering, Chongqing University, China	11
Beijing Institute of Technology, Beijing, China	8
College of Computer Science, National University of Defense Technology, China	5
Department of Computer Science, University of Calgary, Canada	5
Federal University of Pernambuco, Brazil	4
Electrical and Computer Engineering, University of British Columbia, Canada	3
Institute of Computer Science, University of Tartu, Estonia	2
Department of Computer Science, University of Victoria, Canada	2
Department of Statistics, University of Peshawar, Pakistan	1
SUNY Binghamton, USA	1

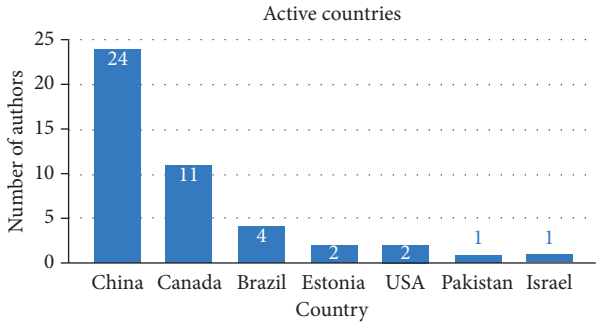

Figure 3: Most active countries (by authors' affiliation).

but better precision/F-measures and was used in only one of the selected primary studies. Besides, among the reviewed ML algorithms, SVM proved to have the excellent performance, with LDA (USL algorithm) being the most recurrently utilized ML algorithm.

The machine learning algorithms seem to perform well when separate words are employed as features requests, rather than the different phrases. Likewise, the ML techniques/algorithms used in the selected studies tend to yield better results when the words used are in the original form, instead of performing some preprocessing steps (stemming and lemmatization, among others) on the words.

The key findings discussed above depict that despite of the fact of being still at the infancy stage of the research, the machine learning based methods have produced considerably good results for identifying and classifying software requirements (FRs/NFRs). This is deemed to be an encouraging prospect.

5.2. SLR Limitations. During this SLR study, we have noticed some limitations specifically related to reporting and evaluating the ML techniques used in the selected research studies discussed as follows.

5.2.1. Deficient Reporting Standards. Predefined data is a needed both for SL and USL algorithms: the SL algorithms need a labelled data, whereas the USL algorithms require predefined classifications and related terms to accomplish better performance. Nonetheless, before employing any machine learning algorithm (the SL or the USL algorithm), the machine learning methods must follow some phases to preprocess the input data (in our case the SO posts) and find the pertinent features for the machine learning algorithm. But, this process is not explicitly stated in the surveyed techniques in the selected research studies of our SLR study. Besides, the majority of the chosen primary research studies treat the machine learning approaches as 'black boxes' and offer no explicit explanation on how actually all these machine learning approaches work. Consequently, this makes the SLR study fairly challenging. To guarantee the consistency of our SLR study, we have made a general process (see Figure 2) as a common framework to evaluate each of the selected studies individually. Moreover, we also thoroughly assessed the reported results of the selected studies to infer how the employed approaches worked.

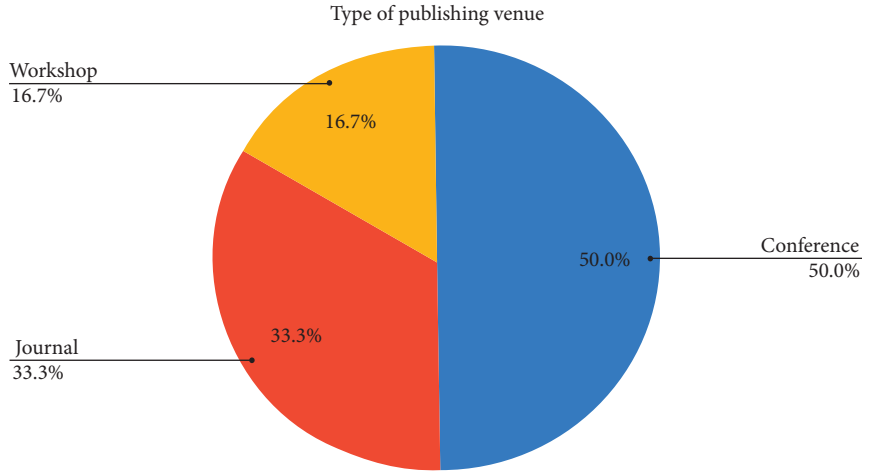

Figure 4: Distribution of publishing venues.

Table 16: Most cited conferences, journals, and workshops.

Publication name	Type of publication	Support level
ACM/IEEE Working Conference on Mining Software Repositories	Conference	3
IEEE International Conference on Software Engineering	Conference	1
ACM International Conference on Communication and Information Processing	Conference	1
Requirements Engineering in the Big Data Era	Conference	1
Information and Software Technology	Journal	1
Empirical Software Engineering	Journal	1
IEEE Access	Journal	1
International Journal of Machine Learning and Computing (IJMLC)	Journal	1
IEEE International Requirements Engineering Conference Workshops	Workshops	1
ACM workshop on Workshop on Refactoring Tools (WRT)	Workshops	1

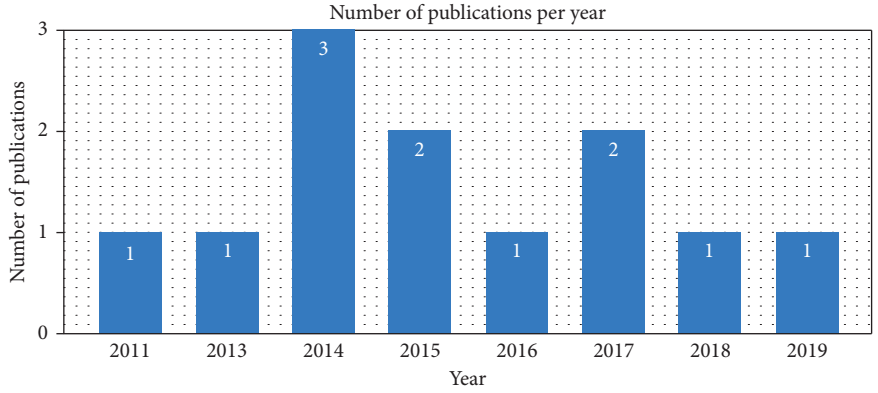

Figure 5: Evolution of the number of papers published.

5.2.2. Deficient Performance Evaluation Standards. In our SLR study, half of the selected research studies gave the evaluation results (6 out of 12 studies); the majority of them did not give the reasons behind using a certain performance evaluation technique. For instance, considering the precision and recall measure, the selected research studies did not entirely clarify which one was more significant: low recall and high precision measures or low precision and high recall measures. Besides, they did not state the reasons behind providing F-measure, word intrusion, and topic intrusion measure and not providing accuracy measure, or vice versa. Thus, we can conclude that the majority of the selected studies of our SLR study did not determine why they have used a certain technique that varies from others or how to describe their outcomes.

5.3. Open Issues. Through thorough assessment and thinking on this SLR study, we finally managed to identify five open issues normally confronted by practitioners or researchers, briefly explained as follows.

5.3.1. Need for Shared Standard Prelabelled Datasets. The first critical issue identified in the existing research is the deficiency of a standard prelabelled dataset. A preclassified standard dataset is mandatory to employ or utilize the SL algorithms more effectively. It considerably needs a lot of efforts and time to develop such a standard dataset. Besides, it needs the pool of real-life FRs/NFRs, the standard classification of these requirements FRs/NFRs, and the validation of the classifications from researchers and practitioners (domain experts). As per our knowledge, unfortunately there are no shared prelabelled datasets available; developing a shared prelabelled dataset can assist the researchers/practitioners to perform more experimentation and offer benchmarks for future performance. Nonetheless, the abundance of such prelabelled dataset will not only solve the problem of automated learning, but also aid in improving the systematic authentication of such applications.

5.3.2. Need for Standardized Nonfunctional Requirements. RE plays a key role in any software project. Specifically, the role of NFRs is considered significant and critical to the ultimate success of any type of software projects [79]; nonetheless, till now the RE community (both researchers and practitioner) could not develop a consensus on what actually nonfunctional requirements (NFRs) are and how effectively can they be classified and represented [79]. Subsequently, there exists a plethora of varied terms (e.g., qualities, attributes, properties, characteristics, constraints, and performance) used to define the nonfunctional requirements, ultimately causing not only terminologies, but also vast conceptual differences [79].

The variety in the definitions of the NFRs ultimately causes different classification of the NFRs. Our SLR results show that only few of the selected studies (e.g., S04 and S08) used the same NFRs categorization while S05 used a different category. Thus, there is a need for standard definition and framework of NFRs which will ultimately assist future researchers to do more consistent experimentation.

Along with the problems of the standardized definition and categorization or classification of the NFRs, the third challenge is the proper NFRs representation: till now there is no standard agreement on how uniformly the NFRs could possibly be described and particularly what precise level of detail is desired [79]. One possible effect is that an NFR might turn into a functional requirement, dependent upon how you describe or express it. For instance, the software requirements example stated by Glinz [79], "the probability for successful access to the customer data by an unauthorized person shall be smaller than 10–5," is actually an NFR. Nevertheless, if we further polish this software requirement to "the database shall grant access to the customer data only to those users that have been authorized by their user name and password," it eventually converts to a functional requirement.

These three aforementioned issues make the different machine learning algorithms quite problematic, as they could possibly be merely trained and utilized for particular NFRs with particular terminologies and categorization. They specifically could not be generalized or scaled up. These issues ultimately make the automated categorization/classification of the software requirements quite difficult and easily disposed to errors. Moreover, these sorts of issues create difficulties for comparing and evaluating the performance of similar methods or techniques and to set up some performance milestones.

5.3.3. Finding and Selecting Useful Features. Mining and extracting precisely the valuable features from the SO posts is a challenging task. The ML approaches utilized by the selected research studies do not necessarily offer significant features. In the USL ML approaches (LDA), the features were either too abstract or worthless. For instance, the abstract features adapted by S06 are, for example, 'failed' and 'fails' for reliability and the features adopted by S08 are, for example, 'user' and 'friend' for usability. Besides, in S06 the "node" and "code" can be deemed as worthless features for the category of the usability. On the other hand, the SL approaches too revealed considerable performance; however, no clear examples of insufficient or inappropriate features were reported which can be discussed here.

It is obvious that worthless features could considerably cause the surge in the number of the false positives and will ultimately lead to the poor performance of the classifiers. Nevertheless, this also hints that the classifiers can be limited in effectively recognizing and extracting NFRs from the domain for which they are trained leading to creating the overfitting problem. We also think that the SL approaches might not be enough capable of bing effectively useful with a diverse style of writing in the similar domains.

Besides, the text of SO posts is considerably in various lengths (shorter or longer) and composition (code written inside texts) compared to the sentences utilized in routine language. These likely cause the lower cooccurrences of the words and the sparse features. Thus, we think that identifying and extracting precisely the meaningful features for NFRs is still an open challenging issue which needs more investigations in the future.

5.3.4. Need for Sentiment Analysis. The selected studies of our SLR study revealed that all the ML approaches (both the SL and USL) and other techniques did not perform any sentiment analysis. For instance, they just found the list of requirements (FRs/NFRs) from the SO posts and did not further analyze the extracted requirements. More specifically, they just treated all the extracted requirements as "black box" and none of the selected studies further classified whether the extracted requirements are feature requests, bug reports, praise, and positive or negative, among others. Thus, one possible way of addressing and performing the sentiment analysis would be adopting the work of [80, 81] for Q&A posts. Nevertheless, another challenging issue that needs to be addressed is investigating the suitability of adopting different sentiment analysis techniques as reported in [82–85] for our domain.

5.3.5. Need for Applying Advanced ML Algorithms. In our SLR study, there were mainly two ML algorithms used LDA (USL) and SVM (SL) for identifying requirements in SO posts. There is a lack of applying not only other SL and SSL algorithms but also advanced ML algorithms. The results and performance of both of these ML algorithms were good enough; however, keeping in view the importance and structure of the text and volume of the data and applying some advanced algorithms (i.e., deep learning) are worth investigating in the future.

6. Validity Threats

In this study, we have considered the five most frequently reported validity threats of software engineering research [60, 86]. These validity threats include descriptive validity, interpretive validity, theoretical validity, generalizability, and reliability. In the subsequent sections, each of these validity threats is discussed in detail.

6.1. Descriptive Validity. This type of validity is associated with the threats to a person's ability to apprehend and document the observations in an objective and accurate way. In SLR, the most significant activity related to this kind of threat is the precise extraction of data since the researchers' biasness may affect the type and quantity of the extracted data. Besides, there might be variations in the interpretation, meaning, and description of the extracted data among researchers. To minimize this threat, the authors arranged work sessions using examples to realize a uniform methodology for determining how and what data need to be extracted. To accomplish this aim, we designed a form called DEF, which was unanimously agreed upon by all the authors. To ensure uniformity and traceability and ultimately reduce the researcher's bias, every entry in the DEF was commented that linked the value assigned by the researcher to a specific text in the original source. Finally, two of the researchers independently reviewed every piece of data extracted, and agreement sessions were arranged whenever there is a discrepancy in the outputs or situations when some descriptive problems arose.

6.2. Interpretive Validity. This kind of validity threats is also called conclusions validity, as they might occur while figuring out any conclusions. Nevertheless, again, the main threat is the researchers' bias when comprehending the extracted data.

To minimize this threat, we applied two different mechanisms. First, regular sessions were arranged after the data extraction, to make sure that all the authors agreed on the correct interpretation of the results, a set of coding rules, and their effects. Secondly, four of the authors in two independent teams acquired the conclusions from the results. Besides, the first author steered, matched, and combined the conclusions, harmonizing the writing style. Finally, all of the authors double-checked the conclusions to ensure that they could be traceable to all of the previous results stored in the DEF.

6.3. Theoretical Validity. This is one of the crucial types of validity threats that contain the highest risk. There are many activities that are mostly affected which are searching and selecting primary papers; nonetheless individual researchers' bias during data extraction (descriptive validity) is also a crucial factor. The main problem related to the search process is the inability to discover all the related available evidence. During the selection process, three challenging situations may occur: selection of the irrelevant papers, or exclusion of the relevant papers, or both. Some other threats are publication bias towards positive results and the quality of the selected works, which ultimately have impact on the data extracted and the acquired conclusions.

To reduce the threat to the search process, we developed the automated search strings from the diverse key terms generated from our defined set of the research questions (RQs), keywords from the research publications retrieved by a pilot search, and the list of various synonyms. Besides, we performed an automated search in four separate electronic database sources. The protocol to perform the search process is detailed in Search Strategy (Section 3.2). The details concerning the selection process and the protocol followed to minimize the theoretical validity threats can be found in Research Paper Selection Criteria (Inclusion and Exclusion Criteria) (Section 3.3).

6.4. Generalizability. There are two types of validity threats that need to be considered under this validity category, namely, internal validity threat and external validity threat. Both of these types are related to the possibility of generalizing the results, within the groups known as internal validity or between the groups known as external validity.

Internal generalizability is related to the validity of our results within other groups identifying software requirements on SO. This validity is reliant on the selected primary paper because of the diversity in reported contexts (type of Q&A, number of Q&A, techniques or algorithms used, domain, assessed metrics, and so on) and the low number of primary studies. It is also dependent upon the fact that the quality of the reported information is also low, due to the scarcity of enough details).

External generalizability is primarily related to other external groups or communities. We did not investigate the identification of software requirements in other settings different from the SO. All of the selected studies deal with identification of software requirements on SO. Henceforth, it is impossible to make judgments regarding other contexts.

To lessen the possible internal generalizability threat, we depend upon the objectivity of the data extraction process and form (DEF) and the SLR protocol to assess the results and acquire significant conclusions. Nonetheless, due to the sample size (12 primary studies), the generalization of the results cannot be guaranteed by our study and needs more investigation.

6.5. Reliability. This kind of the validity threat is concerned about the capability of other researchers to replicate the work and to get similar outcomes. It is challenging to

replicate the experiments in the software engineering domain [41]; nevertheless SLR is an objective mechanism that assists the critical facets of this replication. By providing adequate detailed data on the different information sources (e.g., research databases, search engines, and search strings), well defined exclusion and inclusion criteria, an objective DEF, and granting access to all of the data, comprising those obtained in intermediary processes, the probabilities of other researchers to replicate the study increase.

To improve the reliability of our work, we conducted a comprehensive report of the entire process pursued, from the start of the protocol to the final conduction phase. Lastly, to minimize the validity threats during the conduction phase of this work, we described the rubrics used for the self-evaluation, pursuing the detailed procedures defined by Kitchenham et al. [25, 26].

7. Conclusions

This research study reports comprehensively the planning, conducting, and execution phases of the SLR work on using ML techniques for recognizing diverse software requirements on the SO online Q&A platform. The SLR study was primarily based on the identification, analysis, and classification of 12 selected primary studies that were published until May 2020. The authors have thoroughly investigated several RQs pertaining to what machine learning algorithms/techniques have been effectively utilized for diverse software requirements identification/recognition on the SO, assessed the actual working of these approaches/techniques, and eventually identified the performance measures that have been utilized to assess these techniques or approaches. The authors have thoroughly investigated addressing these RQs and provided a systematic comprehension of the identified machine learning based techniques.

One of the conclusive points that come from the SLR outcomes is that properly using machine learning algorithms to extract or identify software requirements on the SO platform is a very challenging job, as it needs suitable techniques to elicit features and to identify and categorize/classify the text. For the SL machine learning approaches, a prelabelled dataset is compulsory as well. On the other side, comparing with the old-fashioned manual classification methods, using machine learning approaches can possibly support in improving the identification and extraction process of the NFRs, as it decreases the human labours and mistakes. This SLR revealed that machine learning algorithms, specifically SVM (SL algorithm), possess great potential in the RE domain, as they proved to have considerably better performance, LDA (USL algorithm) is the most widely used machine learning algorithm, and the precision and recall are amongst the commonly utilized evaluation methods to measure the performance of these machine learning algorithms in the selected research studies.

Our SLR study, being of an empirical research nature, pointed out several avenues for future researchers, in terms of not only tackling the gaps identified, but also developing new and improved machine learning methods. We have thoroughly discussed the suggestions derived from this SLR study in Key Findings (see Section 5.1), Limitations (see Section 5.2), and some of the avenues for future researchers in the open issues (see Section 5.3).

In conclusion, our SLR findings advocate that this area of research is neglected before and believe that our SLR study is the first step in improving the knowledge base, as it briefly reported the outcomes of all the recent developments related to the RE research on SO. Another interesting finding from our SLR revealed that it appears like this domain is passing through a prescientific phase: all the selected studies analyzed in our SLR seldom compared themselves with the existing works. Finally, we conclude that this SLR study calls for the effective collaboration venture between the RE and machine learning researchers or practitioners to tackle the open challenges confronted in the development of the real world machine learning applications to the enormous field of RE.

Data Availability

The data used to support the findings of the study are included within the manuscript.

Conflicts of Interest

All authors declare that they have no conflicts of interest relevant to this article.

Acknowledgments

This work was funded by the National Key R&D Program of China (no. 2017YFB1002101), National Natural Science Foundation of China (no. U1636203), and the Joint Advanced Research Foundation of China Electronics Technology Group Corporation (CETC) (no. 6141B08010102).

References

[1] H. Khan, A. Ahmad, and M. A. Alnuem, "Knowledge management: a solution to requirements understanding in global software engineering," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 4, pp. 2087–2099, 2012.

[2] H. Khan, A. Ahmad, C. Johansson, and M. A. Alnuem, "Requirements understanding in global software engineering industrial surveys," in *Proceedings of the 2011 International Conference on Computer and Software Modeling (IPCSIT)*, pp. 167–173, Izmir, Turkey, July 2011.

[3] A. Ahmad and H. Khan, "The importance of knowledge management practices in overcoming the global software engineering challenges in requirements understanding," Master thesis research, Blekinge Institute of Technology, Karlskrona, Sweden, 2008.

[4] M. A. Alnuem, A. Ahmad, and H. Khan, "Requirements understanding: a challenge in global software development, industrial surveys in Kingdom of Saudi Arabia," in *Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC)*, pp. 297–306, Izmir, Turkey, July 2012.

[5] A. Ahmad, C. Feng, M. Tao, A. Yousif, and S. Ge, "Challenges of mobile applications development: initial results," in *Proceedings of the 8th IEEE International Conference on Software*

Engineering and Service Science (ICSESS 2017), pp. 464–469, Beijing, China, November 2017.

[6] A. Ahmad, K. Li, C. Feng, S. M. Asim, A. Yousif, and S. Ge, "An empirical study of investigating mobile applications development challenges," *IEEE Access*, vol. 6, pp. 17711–17728, 2018.

[7] J. Dick, E. Hull, and K. Jackson, *Requirements Engineering*, Springer, Berlin, Germany, 2017.

[8] A. Ahmad, "Research on comprehending software requirements on social media," *PhD ComputerScience [amp] Technology Research*, School of Computer Science & Technology, Beijing Institute of Technology, Beijing, China, 2018.

[9] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk, "An exploratory analysis of mobile development issues using stack overflow," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, pp. 93–96, San Francisco, CA, USA, May 2013.

[10] E. C. Groen, S. Kopczýnska, M. P. Hauer, T. D. Krafft, and J. Doerr, "Users – the hidden software product quality experts? A study on how app users report quality Aspects in online reviews," in *Proceedings of the IEEE 25th International Requirements Engineering Conference*, pp. 80–89, Lisbon, Portugal, September 2017.

[11] C. Li, L. Huang, J. Ge, B. Luo, and V. Ng, "Automatically classifying user requests in crowdsourcing requirements engineering," *Journal of Systems and Software*, vol. 138, pp. 108–123, 2018.

[12] Z. Kurtanovic, W. Maalej, and Ieee, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference*, pp. 490–495, Lisbon, Portugal, September 2017.

[13] A. Ahmad, C. Feng, S. Ge, and A. Yousif, "A survey on mining stack overflow: question and answering (Q&A) community," *Data Technologies and Applications*, vol. 52, no. 2, pp. 190–247, 2018.

[14] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider, "What works better? a study of classifying requirements," in *Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 496–501, Lisbon, Portugal, September 2017.

[15] N. A. Ernst and J. Mylopoulos, "On the perception of software quality requirements during the project lifecycle," in *Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 143–157, Essen, Germany, June 2010.

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

[17] F. Petcuşin, L. Stănescu, and C. Bădică, "An experiment on automated requirements mapping using deep learning methods," in *Proceedings of the International Symposium on Intelligent and Distributed Computing*, pp. 86–95, Saint-Petersburg, Russia, October 2019.

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

[19] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Requirements classification with interpretable machine learning and dependency parsing," in *Proceedings of the 2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 142–152, Jeju Island, Korea, September 2019.

[20] Y. Ko, S. Park, J. Seo, and S. Choi, "Using classification techniques for informal requirements in the requirements analysis-supporting system," *Information and Software Technology*, vol. 49, no. 11-12, pp. 1128–1140, 2007.

[21] M. Galster, F. Gilson, and F. Georis, "What quality attributes can we find in product backlogs? A machine learning perspective," in *Proceedings of the European Conference on Software Architecture*, pp. 88–96, Paris, France, September 2019.

[22] A. Ahmad, C. Feng, K. Li, S. M. Asim, and T. Sun, "Toward empirically investigating non-functional requirements of iOS developers on stack overflow," *IEEE Access*, vol. 7, pp. 61145–61169, 2019.

[23] A. Ahmad, L. Kan, C. Feng, and T. Sun, "An empirical study on how iOS developers report quality Aspects on stack overflow," *International Journal of Machine Learning and Computing (IJMLC)*, vol. 8, pp. 501–506, 2018.

[24] Z. S. H. Abad and G. Ruhe, "Using real options to manage technical debt in requirements engineering," in *Proceedings of the 2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pp. 230–235, Ottawa, Canada, August 2015.

[25] B. Kitchenham, *Procedures for Performing Systematic Reviews*, vol. 33, pp. 1–26, Keele University, Keele, UK, 2004.

[26] B. Kitchenham and S. Charters, *Guidelines for Performing Systematic Literature Reviews in Software Engineering*, Keele University and Durham University, Keele, UK, 2007.

[27] C. Pacheco, I. García, and M. Reyes, "Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques," *IET Software*, vol. 12, no. 4, pp. 365–378, 2018.

[28] V. Anu, W. Hu, J. C. Carver, G. S. Walia, and G. Bradshaw, "Development of a human error taxonomy for software requirements: a systematic literature review," *Information and Software Technology*, vol. 103, pp. 112–124, 2018.

[29] D. Dermeval, J. Vilela, I. I. Bittencourt et al., "Applications of ontologies in requirements engineering: a systematic review of the literature," *Requirements Engineering*, vol. 21, pp. 405–437, 2016.

[30] S. Jayatilleke and R. Lai, "A systematic review of requirements change management," *Information and Software Technology*, vol. 93, pp. 163–185, 2018.

[31] A. Baltadzhieva and G. Chrupała, "Question quality in community question answering forums," *ACM SIGKDD Explorations Newsletter*, vol. 17, no. 1, pp. 8–13, 2015.

[32] H. Meth, M. Brhel, and A. Maedche, "The state of the art in automated requirements elicitation," *Information and Software Technology*, vol. 55, no. 10, pp. 1695–1709, 2013.

[33] M. Binkhonain and L. Zhao, "A review of machine learning algorithms for identification and classification of non-functional requirements," *Expert Systems with Applications*, vol. 1, 2019.

[34] T. Iqbal, P. Elahidoost, and L. Lúcio, "A bird's eye view on requirements engineering and machine learning," in *Proceedings of the 2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 11–20, Nara, Japan, December 2018.

[35] A. Yousif, Z. Niu, J. K. Tarus, and A. Ahmad, "A survey on sentiment analysis of scientific citations," *Artificial Intelligence Review*, vol. 52, no. 3, pp. 1805–1838, 2019.

[36] H. Ali, M. N. M. Salleh, K. Hussain et al., "A review on data preprocessing methods for class imbalance problem,"

International Journal of Engineering & Technology, vol. 8, pp. 390–397, 2019.

[37] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information and Software Technology*, vol. 54, no. 1, pp. 41–59, 2012.

[38] D. A. Magües, J. W. Castro, and S. T. Acuna, "HCI usability techniques in agile development," in *Proceedings of the 2016 IEEE International Conference on Automatica (ICA-ACCA)*, pp. 1–7, Curico, Chile, October 2016.

[39] D. A. Magües, J. W. Castro, and S. T. Acuña, "Usability in agile development: a systematic mapping study," in *Proceedings of the 2016 XLII Latin American Computing Conference (CLEI)*, pp. 1–8, Valparaiso, Chile, October 2016.

[40] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. r. Mahrin, "A systematic literature review of software requirements prioritization research," *Information and Software Technology*, vol. 56, no. 6, pp. 568–585, 2014.

[41] B. A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-based software engineering and systematic reviews*, Vol. 4, CRC Press, Boca Raton, FL, USA, 2015.

[42] S. Beecham, D. Bowes, and K.-J. Stol, *Introduction to the EASE 2016 Special Section: Evidence-Based Software Engineering: Past, Present, and Future*, Elsevier, Amsterdam, Netherlands, 2017.

[43] T. Dyba, B. A. Kitchenham, and M. Jorgensen, "Evidence-based software engineering for practitioners," *IEEE Software*, vol. 22, no. 1, pp. 58–65, 2005.

[44] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: an update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[45] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, vol. 12, pp. 1–10, Swindon, UK, June 2008.

[46] J. L. Barros-Justo, F. B. V. Benitti, and S. Tiwari, "The impact of Use Cases in real-world software development projects: a systematic mapping study," *Computer Standards & Interfaces*, vol. 66, p. 103362, 2019.

[47] A. Ahmad, C. Feng, A. Tahir et al., "An empirical evaluation of machine learning algorithms for identifying software requirements on Stack Overflow: initial Results," in *Proceedings of the 10th IEEE International Conference on Software Engineering and Service Science (ICSESS 2019)*, Beijing, China, October 2019.

[48] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, p. 38, London, UK, May 2014.

[49] B. A. Kitchenham, Z. Li, and A. J. Burn, "Validating search processes in systematic literature reviews," in *Proceedings of the 1st International Workshop on Evidential Assessment of Software Technologies (EAST-2011)*, pp. 3–9, Beijing, China, 2011.

[50] J. Bailey, C. Zhang, D. Budgen, M. Turner, and S. Charters, "Search engine overlaps: do they agree or disagree?" in *Proceedings of the Second International Workshop on Realising Evidence-Based Software Engineering*, p. 2, Minneapolis, MN, USA, May 2007.

[51] N. B. Ali and M. Usman, "Reliability of search in systematic reviews: towards a quality assessment framework for the automated-search strategy," *Information and Software Technology*, vol. 99, pp. 133–147, 2018.

[52] L. Chen, M. Ali Babar, and H. Zhang, "Towards an evidence-based understanding of electronic data sources," in *Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering*, London, UK, April 2010.

[53] M. Turner, *Digital Libraries and Search Engines for Software Engineering Research: An Overview*, Keele University, Keele, UK, 2010.

[54] D. Badampudi, C. Wohlin, and K. Petersen, "Experiences from using snowballing and database searches in systematic literature studies," in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, p. 17, Nanjing, China, April 2015.

[55] N. B. Ali and K. Petersen, "Evaluating strategies for study selection in systematic literature studies," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, p. 45, Torino, Italy, September 2014.

[56] K. Petersen and N. B. Ali, "Identifying strategies for study selection in systematic reviews and maps," in *Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement*, pp. 351–354, Banff, Canada, September 2011.

[57] M. Niazi, S. Mahmood, M. Alshayeb et al., "Challenges of project management in global software development: a client-vendor analysis," *Information and Software Technology*, vol. 80, pp. 1–19, 2016.

[58] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: a systematic review," *Information and Software Technology*, vol. 50, no. 9-10, pp. 833–859, 2008.

[59] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard et al., "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721–734, 2002.

[60] A. Ampatzoglou, S. Bibi, P. Avgeriou, M. Ver-beek, and A. Chatzigeorgiou, "Identifying, categorizing and mitigating threats to validity in software engineering secondary studies," *Information and Software Technology*, vol. 106, pp. 201–230, 2018.

[61] D. Budgen, P. Brereton, S. Drummond, and N. Williams, "Reporting systematic reviews: some lessons from a tertiary study," *Information and Software Technology*, vol. 95, pp. 62–74, 2018.

[62] M. Kuhrmann, D. M. Fernández, and M. Daneva, "On the pragmatic design of literature studies in software engineering: an experience-based guideline," *Empirical Software Engineering*, vol. 22, no. 6, pp. 2852–2891, 2017.

[63] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised Learning*, MIT Press, Cambridge, MA, USA, 2006.

[64] D. M. Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[65] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[66] V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," in *Measures of Complexity*, pp. 11–30, Springer, Berlin, Germany, 2015.

[67] R. Afridi, Z. Iqbal, M. Khan, A. Ahmad, and R. Naseem, "Fetal heart rate classification and comparative analysis using cardiotocography data and KNOWN classifiers," *International Journal of Grid and Distributed Computing (IJGDC)*, vol. 12, pp. 31–42, 2019.

[68] J. Fereday and E. Muir-Cochrane, "Demonstrating rigor using thematic analysis: a hybrid approach of inductive and deductive coding and theme development," *International Journal of Qualitative Methods*, vol. 5, no. 1, pp. 80–92, 2006.

[69] G. A. Bowen, "Document analysis as a qualitative research method," *Qualitative Research Journal*, vol. 9, no. 2, pp. 27–40, 2009.

[70] V. Braun and V. Clarke, *Thematic Analysis*, Springer, Berlin, Germany, 2012.

[71] A. Khan, B. Baharudin, L. H. Lee, and K. Khan, "A review of machine learning algorithms for text-documents classification," *Journal of Advances in Information Technology*, vol. 1, pp. 4–20, 2010.

[72] A. G. Jivani, "A comparative study of stemming algorithms," *International Journal of Computer Applications in Technology*, vol. 2, pp. 1930–1938, 2011.

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

[74] A. Casamayor, D. Godoy, and M. Campo, "Identification of non-functional requirements in textual specifications: a semi-supervised learning approach," *Information and Software Technology*, vol. 52, no. 4, pp. 436–445, 2010.

[75] W. Zhang, Y. Yang, Q. Wang, and F. Shu, "An empirical study on classification of non-functional requirements," in *Proceedings of the Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011)*, pp. 190–195, Miami Beach, FL, USA, July 2011.

[76] M. Riaz, J. King, J. Slankas, and L. Williams, "Hidden in plain sight: automatically identifying security requirements from natural language artifacts," in *Proceedings of the 2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pp. 183–192, Karlskrona, Sweden, August 2014.

[77] D. Mladenić, J. Brank, M. Grobelnik, and N. Milic-Frayling, "Feature selection using linear classifier weights: interaction with classification models," in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 234–241, Sheffield, UK, July 2004.

[78] J. Chang, S. Gerrish, C. Wang, J. L. Boyd-Graber, and D. M. Blei, "Reading tea leaves: how humans interpret topic models," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 288–296, Vancouver, BC, Canada, December 2009.

[79] M. Glinz, "On non-functional requirements," in *Proceedings of the 15th IEEE International Requirements Engineering Conference (RE 2007)*, pp. 21–26, Delhi, India, October 2007.

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

[81] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *Proceedings of the 2015 IEEE 23rd international requirements engineering conference (RE)*, pp. 116–125, Ottawa, Canada, August 2015.

[82] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto, "Sentiment analysis for software engineering: how far can we go?" in *Proceedings of the 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pp. 94–104, Gothenburg, Sweden, May 2018.

[83] D. M. E.-D. M. Hussein, "A survey on sentiment analysis challenges," *Journal of King Saud University - Engineering Sciences*, vol. 30, no. 4, pp. 330–338, 2018.

[84] N. Novielli, D. Girardi, and F. Lanubile, "A benchmark study on sentiment analysis for software engineering research," in *Proceedings of the 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*, pp. 364–375, Gothenburg, Sweden, May 2018.

[85] H. Tang, S. Tan, and X. Cheng, "A survey on sentiment detection of reviews," *Expert Systems with Applications*, vol. 36, no. 7, pp. 10760–10773, 2009.

[86] K. Petersen and C. Gencel, "Worldviews, research methods, and their relationship to validity in empirical software engineering research," in *Proceedings of the 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*, pp. 81–89, Ankara, Turkey, October 2013.