SN Computer Science (2021) 2:16
https://doi.org/10.1007/s42979-020-00416-4

REVIEW ARTICLE

Data-Driven Requirements Elicitation: A Systematic Literature Review

Sachiko Lim[1] · Aron Henriksson[1] · Jelena Zdravkovic[1]

Received: 25 March 2020 / Accepted: 2 December 2020

Abstract
Requirements engineering has traditionally been stakeholder-driven. In addition to domain knowledge, widespread digitalization has led to the generation of vast amounts of data (Big Data) from heterogeneous digital sources such as the Internet of Things (IoT), mobile devices, and social networks. The digital transformation has spawned new opportunities to consider such data as potentially valuable sources of requirements, although they are not intentionally created for requirements elicitation. A challenge to data-driven requirements engineering concerns the lack of methods to facilitate seamless and autonomous requirements elicitation from such dynamic and unintended digital sources. There are numerous challenges in processing the data effectively to be fully exploited in organizations. This article, thus, reviews the current state-of-the-art approaches to data-driven requirements elicitation from dynamic data sources and identifies research gaps. We obtained 1848 hits when searching six electronic databases. Through a two-level screening and a complementary forward and backward reference search, 68 papers were selected for final analysis. The results reveal that the existing automated requirements elicitation primarily focuses on utilizing human-sourced data, especially online reviews, as requirements sources, and supervised machine learning for data processing. The outcomes of automated requirements elicitation often result in mere identification and classification of requirements-related information or identification of features, without eliciting requirements in a ready-to-use form. This article highlights the need for developing methods to leverage process-mediated and machine-generated data for requirements elicitation and addressing the issues related to variety, velocity, and volume of Big Data for the efficient and effective software development and evolution.

Keywords Requirements engineering · Requirements elicitation · Big Data · Automation

Introduction

Requirements elicitation is one of the most critical activities in requirements engineering, which, in turn, is a major determinant of successful development of information systems [1]. In conventional requirements engineering, requirements are elicited from domain knowledge obtained from stakeholders, relying primarily on qualitative data collection methods (e.g., interviews, workshops, and focus group discussions) [2]. The ongoing digitalization of organizations and society at large—as seen, for instance, by the proliferation of e-commerce and the advent of IoT—has led to an unprecedented and increasing amount of high-velocity and heterogeneous data, which is often referred to as Big Data [3].

The digital transformation has spawned new opportunities to consider this type of dynamic data from digital sources as potentially valuable sources of requirements, in addition to domain knowledge. Harnessing both traditional and new data sources in a complementary fashion may help improve the quality of existing or facilitate the development of new software systems. Nevertheless, conventional elicitation techniques are often time-consuming and not sufficiently scalable for processing such fast-growing data or capable of considering stakeholder groups that are becoming

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s42979-020-00416-4.

✉ Sachiko Lim
sachiko@dsv.su.se

Aron Henriksson
aronhen@dsv.su.se

Jelena Zdravkovic
jelenaz@dsv.su.se

[1] Department of Computer and Systems Sciences, Stockholm University, DSV, PO Box 7003, 164 07 Kista, Stockholm, Sweden

Published online: 04 January 2021

increasingly large and global. This highlights the need for a data-driven approach to support continuous and automated requirements engineering from ever-growing amounts of data.

There have been numerous efforts to automate requirements elicitation from static data, i.e., data that are generated with a relatively low velocity and rarely updated. These efforts can be grouped according to the following three aims: (1) eliciting requirements from static domain knowledge (e.g., documents written in natural languages [4, 5], ontologies [6, 7], and various types of models, e.g., business process models [8], UML use cases and sequence diagrams [9]), (2) performing specific requirements engineering activities based on requirements that have been already elicited (e.g., requirements prioritization [10], classification of natural language requirements [11], management of requirements traceability [12], requirements validation [13], generation of a conceptual model from natural language requirements [14]), or (3) developing tools to enhance stakeholders' ability to perform requirements engineering activities based on static domain knowledge or existing requirements (e.g., tool-support for collaborative requirements prioritization [15] and requirements negotiation with rule-based reasoning [16]).

Several systematic reviews have been conducted on automated requirements elicitation from static domain knowledge. Meth et al. conducted a systematic review on tool support for automated requirements elicitation from domain documents written in natural language, where they analyzed and categorized the identified studies according to an analytical framework which consists of tool categories, technological concepts, and evaluation approaches [17]. Nicolás and Toval conducted a systematic review of the methods and techniques for transforming domain models (e.g., business models, UML models, and user interface models), use cases, scenarios, and user stories into textual requirements [18]. In both of these reviews, the requirements sources contained static domain knowledge.

Much less focus has been placed on eliciting requirements from dynamic data, and data that were not intentionally collected for the purpose of requirements elicitation. There are four main advantages to focus on dynamic data from such "unintended" digital sources. First, dynamic data-driven requirements engineering facilitates secondary use of data, which eliminates the need for collecting data specifically for requirements engineering, in turn enhancing scalability. Second, unintended digital sources can include data relevant for new system requirements that otherwise would not be discovered since utilizing such data sources allows for the collection of data from larger and global stakeholders who are beyond the reach of an organization relying on traditional elicitation methods [19]. Including such requirements, which a current software system is not supporting, can bring business values in the form of improved customer satisfaction, cost and time reduction, and optimized operations [20]. Third, focusing on dynamic data allows for capturing up-to-date user requirements, which in turn enables timely and effective operational decision making. Finally, dynamic data from unintended digital sources are machine-readable, which facilitates automated and continuous requirements engineering. A fitting requirements elicitation approach provides new opportunities and competitive advantages in a fast-growing market by extracting real-time business insights and knowledge from a variety of digital sources.

Crowd-based requirements engineering (CrowdRE) is a good example that has taken advantage of dynamic data from unintended digital sources. A primary focus of CrowdRE has been on eliciting requirements from explicit user feedback from crowd users (e.g., app reviews and data from social media) by applying various techniques based on machine learning and natural language processing [21]. Genc-Nayebi and Abran conducted a systematic review on opinion mining from mobile app store user reviews to identify existing solutions and challenges for mining app reviews, as well as to propose future research directions [22]. They focused on specific data-mining techniques used for review analysis, domain adaptation methods, evaluation criteria to assess the usefulness and helpfulness of the reviews, techniques for filtering out spam reviews, and application features. Martin et al. [26] surveyed on studies that performed app store analysis to extract both technical and non-technical attributes for software engineering. Tavakoli et al. [27] conducted a systematic review on techniques and tools for extracting useful software development information through mobile app review mining. The aforementioned literature reviews only focus on utilizing app reviews, while leaving out other types of human-sourced data that are potentially useful as requirement sources. There is also a growing interest in embracing contextual and usage data of crowd users (i.e., implicit user feedback) for requirements elicitation. This systematic review, thus, broadens the scope of previous literature reviews by considering more diverse data sources than merely app reviews for requirements elicitation.

Another relevant approach to data-driven requirements engineering is the application of process mining capabilities for requirements engineering. Process mining is an evidence-based approach to infer valuable process-related insights primarily from event logs, discovered models, and pre-defined process models. Process mining can be divided into three types: process discovery, conformance checking, and process enhancement [23]. Ghasemi and Amyot performed a systematic review on goal-oriented process modeling in which the selected studies were categorized into three areas: (1) goal modeling and requirements elicitation, (2) intention mining (i.e., the discovery of intentional process models going beyond mere activity process models), and (3) key performance indicators (i.e., means for monitoring goals)

[23]. Their findings indicate that the amount of research on goal-oriented process mining is still limited. In addition to explicit and implicit user feedback, as well as event logs and process models, there may be more opportunities to leverage a broader range of dynamic data sources for requirements engineering, such as sensor readings.

Zowghi and Coulin [24] performed a comprehensive survey on techniques, approaches, and tools used for requirements elicitation. However, their work exclusively focused on conventional, stakeholder-driven requirements elicitation methods. Our study instead investigated the data-driven requirements elicitation. More recently, Arruda and Madhavji [25] systematically reviewed the literature on requirements engineering to develop Big Data applications. They identified the process and type of requirements needed for developing Big Data applications, identified challenges associated with requirements engineering in the context of Big Data applications, discussed the available requirements engineering solutions for the development of Big Data applications, and proposed future research directions. This study is different from their work because we studied methods to elicit requirements from Big Data rather than eliciting requirements for Big Data applications.

To our knowledge, no systematic review has been performed with an explicit focus on automated requirements elicitation for information systems from three types of dynamic data sources: human-sourced data sources, process-mediated data sources, and machine-generated data sources. The aim of this study is, therefore, to perform a comprehensive and systematic review of the research literature on existing state-of-the-art methods for facilitating automatic requirements elicitation for information systems driven by dynamic data from unintended digital sources.

This review may help requirements engineers and researchers understand the existing data-driven requirements elicitation techniques and gaps need to be addressed to facilitate data-driven requirements elicitation. Those insights may provide a basis for further development of algorithms and methods to leverage the increasing availability of Big Data as requirements sources.

Definitions and Scope

In this study, *dynamic data* are defined as raw data available in a digital form that changes frequently and have not already been analyzed or aggregated. Dynamic data certainly include but are not limited to Big Data, which in itself is challenging to define [28]. In addition to Big Data, dynamic data also include data that does not strictly meet the 4 Vs of Big Data (i.e., Volume, Variety, Veracity, and Velocity) but are still likely to contain relevant requirements-related information. Domain knowledge includes, for example, intellectual property, business documents, existing system specifications, goals, standards, conferences, and knowledge from customers or external providers.

This study excludes static domain knowledge that is less frequently created or modified and has been the primary focus of existing automated requirements engineering. *Unintended digital sources* are defined as sources of data generated via digital technologies that are unintended with respect to requirements elicitation. Thus, *dynamic data from unintended digital sources* are the digital data pulled from data sources that are created/modified frequently without the intention of eliciting requirements.

Of note is that the two terms “dynamic data” and “unintended digital source” together define the scope of this systematic review. For example, although domain documents are often created without the intention of performing requirements engineering, they are not considered to be dynamic data and, therefore, outside of the scope of this study.

Dynamic data from unintended digital sources expand explicit and implicit user feedback, defined by Morales-Ramirez et al. [29]. In their study, user feedback is considered as “a reaction of users, which roots in their perceived quality of experience”, which indicates the existence of a specific user is assumed. However, there are many devices which collect Big Data such as environmental IoT sensors to measure temperature, humidity, and pollution level, without interacting users. Since we foresee the possibility of eliciting requirements from such data sources, we decided to use a different term from the term “implicit user feedback”. To categorize the sources of data, we used human-sourced, process-mediated, and machine-generated data, following Firmani et al. [30].

Research Questions

To achieve the aim of the study, we formulated the main research question as follows: how can requirements elicitation from dynamic data be supported through automation? The main research question has been further divided into the following sub-research questions:

- RQ1: What types of dynamic data are used for automated requirements elicitation?
  - We focus on describing the sources of the data, but also study whether there have been attempts to integrate multiple types of data sources and whether domain knowledge has been used in addition to dynamic data.
- RQ2: What types of techniques and technologies are used for automating requirements elicitation?
  - We are interested in learning which underlying techniques and technologies are used in the proposed

  - methods, as well as how they are put together and evaluated.
- RQ3: What are the outcomes of automated requirements elicitation?
  - We assess how far the proposed methods go in automating requirements elicitation, the form of the outputs generated by the data-driven elicitation method, and what types of requirements are elicited.

This systematic review will advance scientific knowledge on data-driven requirements engineering for continuous system development and evolution by (1) providing a holistic analysis of the state-of-the-art methods that support automatic requirements elicitation from dynamic data, (2) identifying associated research gaps, and (3) providing directions for future research. The paper is structured as follows: the second section presents the research methods used in our study; the third section presents an overview of the selected studies and the results based on our analytical framework; the fourths section provides a detailed analysis and discussion of each component of the analytical framework; the fifth section describes potential threats to validity; finally, the last section concludes the paper and suggests directions for future work.

Methods

A systematic literature review aims to answer a specific research question using systematic methods to consolidate all relevant evidence that meets pre-defined eligibility criteria [3]. It consists of three main phases: planning, conducting, and reporting the review. The main activities of the planning phase are problem formulation and protocol development. Before the actual review process started, we formulated research questions. The study protocol was then developed, conforming to the guideline of the systematic literature review proposed by Kitchenham and Charters [31]. The protocol included the following contents: background, the aim of the study, research questions, selection criteria, data sources (i.e., electronic databases), search strategy, data collection, data synthesis, and the timeline of the study. The protocol was approved by the research group, which consists of the first author and two research experts: one expert in requirements engineering and one expert in data science. The actual review process starts during the conducting phase. The phase includes the following activities: identifying potentially eligible studies based on title, abstract and keywords, selecting eligible studies through full-text screening, extracting and synthesizing data that are relevant to answer the defined research question(s), performing a holistic analysis, and interpreting the findings. During the reporting phase, the synthesized findings are documented and disseminated to an appropriate channel.

Selection Criteria

Inclusion and exclusion criteria were developed to capture the most relevant articles for answering our research questions.

Inclusion Criteria

We included articles that met all the following inclusion criteria:

- Requirements elicitation is supported through automation.
- Requirements are elicited from digital and dynamic data sources.
- Digital and dynamic data sources are created without intention with respect to requirements engineering.
- Changes in requirements should involve the elicitation of new requirements.
- The article has been peer-reviewed.
- The full text of the article is written in English.

Exclusion Criteria

We excluded articles that met at least one of the following exclusion criteria:

- Requirements are elicited solely from non-dynamic data.
- The proposed method is performed based on existing requirements.
- Studies that merely presented the proposed artifact without any or sufficient descriptions of evaluation methods.
- Review papers, keynote talks, or abstracts of conference proceedings.

Data Sources

We performed a comprehensive search in six electronic databases (Table 1). In the first iteration, we searched Scopus, Web of Science, ACM Digital Library, and IEEE Xplore. Those databases were selected because they together cover the top ten information systems journals and conferences [17]. In addition, EBSCOhost and ProQuest, which are two major databases in the field of information systems, were searched to maximize the coverage of relevant publications, in line with a previous systematic

Table 1 Data sources

Electronic databases	Search field	Date of search
Scopus	Title, abstract, keywords	2018-12-05
Web of Science	Topic	2018-12-05
ACM Digital Library	Title, author, abstracts, citations, and keywords	2018-12-05
IEEE Xplore	Metadata (i.e., abstract, title text, and indexing terms)	2018-12-05
EBSCOhost	Subject, keywords, title, abstract	2018-12-21
ProQuest	Anywhere except full text	2018-12-21

review in the area [17]. ProQuest and EBSCOhost include both peer-reviewed and non-peer-reviewed articles. We, however, considered only peer-reviewed articles to be consistent with our inclusion criteria. The differences in the search field across databases are due to the different search functionalities of each electronic database.

Search Strategy

A comprehensive search strategy was developed in consultation with a librarian and the two co-authors who are experts in the fields of requirements engineering and data science, respectively. First, we extracted three key components from the first research question: requirements elicitation, automation, and Big Data sources and related analytics (Table 2). These components formed the basis for creating a logical search string. Big Data can refer either to data sources or to analytics/data-driven techniques to process Big Data. The term is also closely related to data-mining/machine-learning/data science/artificial intelligence techniques. We thus included keywords and synonyms that cover both Big Data sources and related analytics.

To construct a search string, keywords and synonyms that were grouped in the same component were connected by OR-operators, while each key component was connected by AND-operators, which means at least one keyword from each component must be present. The search string was adapted using the specific syntax of each database's search function. The search string was iteratively tested and refined to optimize search results through trial search.

Study Selection

The entire search was performed by the first author (SL). Before starting the review process, we tested a small number of articles to establish agreement and consistency among reviewers. We then conducted a pilot study in which three reviewers independently assessed 50 randomly selected papers to estimate the sample size that is needed to ensure a substantial level of agreement (i.e., 0.61–0.80) based on the Landis and Koch-Kappa's benchmark scale [32]. Each paper was screened by assessing its title, abstract, and keywords against our selection criteria (level 1 screening). During level 1 screening, articles were classified into one of the three categories: (1) included, (2) excluded, or (3) uncertain. Studies that fell into category 1 and 3 proceeded to full-text screening (level 2 screening) since the aim of the level 1 screening was to identify potentially relevant articles or those that lack sufficient information to be excluded.

After each reviewer had assessed 50 publications, we computed the Fleiss's Kappa to calculate the inter-rater reliability. We, however, did not discuss the results of each reviewer's assessment. The Fleiss's Kappa was used because there were more than two reviewers. The Fleiss' Kappa was computed to be 0.786. Sample size estimation was

Table 2 Search terms

Key components	Keywords and synonyms
Requirements elicitation	"Requirements elicitation" OR "requirements analysis" OR "requirements identification" OR "requirements discovery" OR "requirements gathering" OR "requirements determination" OR "requirements collection" OR "requirements engineering" OR "system requirements"
Automation	Automat* OR "computer aided" OR "computer assisted"
Big Data sources and related analytics	"Big data" OR sensor* OR "Internet of Things" OR IoT OR "natural language processing" OR "data mining" OR "artificial intelligence" OR "data processing" OR "data science" OR "data analysis" OR "machine learning" OR "data driven" OR "data oriented" OR "graph analytics"

performed, following a confidence interval approach suggested by Rotondi and Donner [33]. Using 0.786 as the point estimate of Kappa and 0.61 as the expected lower bound, the required minimum sample size was estimated to be 139. The value of 0.61 was used as the lower bound of Kappa because it is the lower limit of "substantial" inter-rater reliability based on the Landis and Koch-Kappa's benchmark scale [32], which is what we had aimed for. Since we achieved a substantial level of agreement and did not discuss results not to influence each other's decisions, each of three reviewers independently continued to screen the remainder of the 89 randomly chosen publications based on titles, abstracts, and keywords (level 1 screening). The overall Fleiss' Kappa for reviewing 139 articles was 0.850, which indicates an "almost perfect" agreement, according to the benchmark scale proposed by Landis and Koch [32]. Since we were able to achieve a very high inter-rater reliability, the rest of the level 1 screening was conducted by a single reviewer (SL). However, all of the three reviewers discussed and reached a consensus on the articles which SL classified as uncertain or could not decide on with sufficient confidence.

Before conducting the level 2 screening, we discussed which information should be extracted from the eligible articles. Based on the discussion, we developed a preliminary analytical framework to standardize the information to be extracted. We tested this on a small number of full-text papers and refined the data extraction form accordingly. In the level 2 screening, at least two authors reviewed the full-text of each paper that has been identified in level 1 screening to assess its eligibility in the final analysis. In addition to keyword-based search on the databases, we also performed forward/backward reference searching of all the included studies. SL extracted data from all the eligible studies, while each of AH and JZ divided the data extraction task by half. This was done to ensure that data extracted by SL could be cross-checked by at least one of the two reviewers who have richer experiences and knowledge. Any disagreements between the two reviewers were referred to the third reviewer and resolved by consensus.

To update the search results, an additional search was performed on July 3, 2020, using the same search query and introducing the two-level screening process (i.e., keyword-based search followed by full-text screening). While filtering can be performed by specifying the publication date and year in some databases, in other databases, the search can only be filtered by the publican year. Thus, we manually excluded the studies that have been published before the date of the initial search. However, we did not perform a backward and forward reference search during the updating phase. We then applied the same selection criteria used for the initial search to identify the relevant studies.

Analytical Framework and Data Collection

After considering the selected articles, we iteratively developed and refined an analytical framework, which covers both design and evaluation perspectives, to answer our research questions. The framework consists of three components: types of dynamic data sources used for automated requirements elicitation, techniquesand technologies used for automated requirements elicitation, and the outcomes of automated requirements elicitation. Table 3 summarizes the extracted data that are associated with each component of the analytical framework. Each component of the analytical framework is described in detail below.

Types of Dynamic Data Sources Used for Automated Requirements Elicitation

To answer RQ1, we extracted the following information: (1) types of dynamic data sources, (2) types of dynamic data, (3) integration of data sources, (4) relation of dynamic data to a given organization, and (5) additional domain knowledge that is used to elicit system requirements.

Types of Dynamic Data Sources Dynamic data sources are categorized into one or a combination of human-sourced data sources, process-mediated data sources, and machine-generated data sources [30]. This provides insights into which types of data sources have drawn the most or the least attention as potential requirements sources in the existing literature. The categorization also helps to analyze whether there exists any process pattern in the automated requirements elicitation within each data source type.

Human-sourced data sources refer to the digitized records of human experiences. To name a few, examples of human-sourced data sources include social media, blogs, and contents from mobile phones. Process-mediated data sources are records of business processes and business events that are monitored, which includes electronic health records, commercial transactions, banking records, credit card payments. Machine-generated data sources are the records of fixed and mobile sensors and machines that are used to measure the events and situations in the physical world. They include, for example, readings from environmental and barometric pressure sensors, outputs of medical devices, satellite image data, and location data such as RFID chip readings and GPS outputs.

Types of Dynamic Data To understand what types of dynamic data have been used for eliciting system requirements in the existing literature, we extracted the specific types of dynamic data that were used in each of the selected studies and grouped them into seven categories. Those cate-

Table 3 Analytical framework

Components	Extracted data	Definition/description
Overview of the study	General characteristics	The number that is uniquely associated with the title, author (s), the name of journal/conference, and year of publication
Types of dynamic data sources used for automated requirements elicitation	Types of dynamic data sources	• Human-sourced data sources • Process-mediated data sources • Machine-generated data sources
	Types of specific dynamic data	• Online reviews • Microblogs • Online discussions/forums • Software repositories • Usage data • Sensor readings • Mailing lists
	Integration of data sources	• Yes • No
	Relation of dynamic data to an organization of interest	• Internal • External
	Additional use of domain knowledge	• Yes • No
Techniques used for requirements elicitation	Techniques(s) used for automation	• Machine learning • Rule-based classification • Model-oriented approach • Topic modeling • Traditional clustering
	Summarization/aggregation	• Yes • No
	Visualization	• Yes • No
	Intended degree of automation	• Full-automation • Semi-automation
	Evaluation approach	• Controlled experiment • Case study • Proof of concept • Other concepts
	Evaluation concepts	• Completeness • Correctness • Efficiency • Other evaluation concepts
	Evaluation metrics	Metrics that were used to evaluate a selected concept(s)
The outcomes of automated requirements elicitation	Expression of requirements	• Identification and classification of requirements-related information • Identification of candidate features related to requirements • Elicitation of requirements
	Additional requirements engineering activity supported through automation	• Yes • No

gories are online reviews (e.g., app reviews, expert reviews, and user reviews), micro-blogs (e.g., Twitter), online discussions/forums, software repositories (e.g., issue tracking systems and GitHub), usage data, sensor readings, and mailing lists.

Integration of Data Sources We explored whether the study integrates multiple types of dynamic data sources (i.e., any combination of human-sourced, process-mediated, and machine-generated data sources). We classified the selected studies into “yes” if the study has used multiple dynamic data sources, otherwise into “no.”

Relation of Dynamic Data to a Given Organization Understanding whether requirements are elicited from external or internal data sources in relation to a given organization is important for requirements engineers to identify potential sources that can bring innovations into the requirements engineering process and facilitate software evolution and development of new promising software systems. We thus classified the selected studies into "yes" if the platform is owned by the organization and "no" if it is owned by a third party.

Additional Domain Knowledge that was Used to Elicit System Requirements We assessed whether the study uses any domain knowledge in combination with dynamic data to explore the possible ways of integrating both dynamic data and domain knowledge. The selected studies were classified into "yes," if the study uses any domain knowledge in addition to dynamic data, otherwise classified into "no."

Techniques Used for Automated Requirements Elicitation

To answer RQ2, the following four types of information were extracted: (1) technique(s) used for automated requirements elicitation, including process pattern of automating requirements elicitation, (2) use of aggregation/summarization, (3) use of visualization, and (4) evaluation methods.

Technique(s) Used for Automation Implementing promising algorithms is a prerequisite for effective and efficient automation of the requirements elicitation process. To identify the state-of-the-art algorithms, specific methods that were used for automating requirements elicitation were extracted and categorized into machine learning, rule-based classification, model-oriented approach, topic modeling, and traditional clustering.

Aggregation/Summarization Summarization helps navigate requirements engineers to pinpoint the relevant information efficiently out of the ever-growing amount of data. We thus assessed whether the study summarizes/aggregates requirements-related information to obtain high-level requirements. If summarization/aggregation is performed, we also extracted specific techniques used for summarization/aggregation.

Visualization Visualization facilitates requirements engineers to interpret the results of data analysis efficiently and effectively as well as to gain (new) insights in data. We assessed whether the study visualizes the output of the study to enhance their interpretability. If visualization is provided, the specific method used for visualization was also extracted.

Evaluation Methods To understand how rigorously the performance of the proposed artifact was evaluated, we extracted methods that were used to assess the artifact. Evaluation methods were further divided into two dimensions: evaluation approach and evaluation concepts and metrics [17]. The evaluation concept of each selected study was categorized into one of the following groups: controlled experiment, case study, proof of concept, and other concepts. In a controlled experiment, the proposed artifact is evaluated in a controlled environment [34]. A case study aims to assess the artifact in-depth in a real-world context [34]. A proof of concept is defined as a demonstration of the proposed artifact to verify its feasibility for a real-world application. Other concepts refer to studies using other approaches to evaluate their artifact that does not fall into any category of the aforementioned evaluation approach. We also extracted evaluation concepts and metrics used for the artifact evaluation. Evaluation concepts were classified into one or more of the following categories: completeness, correctness, efficiency, and other evaluation concepts.

The Outcomes of Automated Requirements Elicitation

To answer RQ3, we assessed the outcomes of automated requirements elicitation by extracting the following information: (1) types of requirements, (2) expression of the elicited requirements (i.e., in what form outputs that were generated by automated requirements elicitation were expressed), and (3) additional requirements engineering activity supported through automation.

Expression of the Elicited Requirements To understand how the obtained requirements are expressed and how far the elicitation activity reached, outputs of automated requirements elicitation were extracted, which were grouped into the following categories: identification and classification of requirements-related information, identification of candidate features related to requirements, and elicitation of requirements.

Intended Degree of Automation Based on the degree of the proposed automated method, the selected studies were classified into either full automation or semi-automation. We classified the study as full automation if the study fulfilled either of the following conditions: (1) the proposed artifact automated the entire requirements elicitation process without human interaction, or (2) the proposed artifact only supports the partial process of requirements elicitation; however, the part it addressed was fully automated. Semi-automation refers to having a human-in-the-loop for automating requirements elicitation, thus requirements are directed by human interactions.

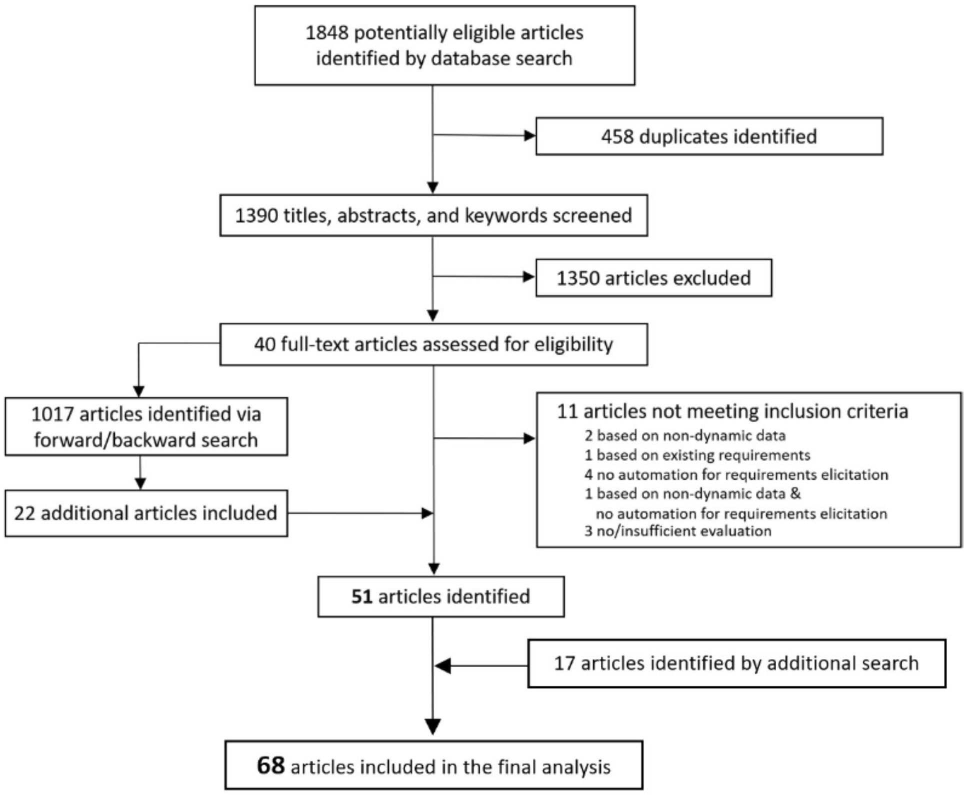
Additional Requirements Engineering Activity Supported Through Automation Understanding to what extent the entire requirements engineering process has already been automated is essential to clarify the direction of future research that aims at increasing the level of automation in performing the requirements engineering process. We thus extracted the requirements engineering activity that was supported through automation other than requirements elicitation, if any.

Quality Assessment

We simply assessed the quality of the selected studies based on CORE Conference Rankings for conferences, workshops, and symposia, and SCImago Journal Rank (SJR) indicators for journal papers. We assumed that a study with a higher score of CORE or SJR has higher quality than one with a lower score. The papers that have been ranked A*, A, B, or C for the CORE index get the point of 1.5, 1.5, 1, and 0.5, respectively. If a paper is ranked Q1 or Q2 for the SJR indicator, the paper receives 2 and 1.5, respectively, while a paper that is ranked Q3 or Q4 gets 1. If a conference/journal paper is not included in the CORE/SJR ranking, the paper scores 0 points.

Data Synthesis

We narratively synthesized the findings of this systematic review, which includes basic descriptive statistics and qualitative analyses of (semi-)automated elicitation methods that are sub-grouped by dynamic data source as well as identified research gap(s), and implications and recommendations for future research.



1848 potentially eligible articles identified by database search
458 duplicates identified
1390 titles, abstracts, and keywords screened
1350 articles excluded
40 full-text articles assessed for eligibility
1017 articles identified via forward/backward search
22 additional articles included
11 articles not meeting inclusion criteria
2 based on non-dynamic data
1 based on existing requirements
4 no automation for requirements elicitation
1 based on non-dynamic data & no automation for requirements elicitation
3 no/insufficient evaluation
51 articles identified
17 articles identified by additional search
68 articles included in the final analysis

Fig. 1 Flow diagram of article selection

Table 4 The results of study identification for each electronic database

Results	Database						Duplicates identified (via Zotero)	Sub-total
	Scopus	Web of science	IEEE Xplore	ACM digital library	EBSCOhost	Proquest		
Initial search	686	508	347	94	201	12	458	1390
Additional search	236	112	101	12	10	2	473	401

Results

Figure 1 shows a flow diagram of the article selection. We obtained 1,848 hits when searching the 6 electronic databases. We removed 458 duplicates, leaving 1,390 articles for level 1 screening (Table 4). After level 1 screening, we identified 40 articles to proceed to level 2 screening. The level 2 screening resulted in the inclusion of 29 articles for data extraction. We excluded the remaining eleven papers due to: the study not using dynamic data for requirements elicitation; the study being based on existing requirements that had already been elicited; the study not automating requirements elicitation to any degree; and the study proposing a method for automated requirements elicitation without sufficient evaluation.

In addition, a forward and backward reference search identified 1017 additional articles. Out of these, 22 articles met our inclusion criteria. Thus, a total of 51 papers were considered in the final analysis. Reasons for similar numbers of articles being identified in the query-based search and the backward/forward search include: the studies using terms such as "elicit requirements", "requirements", "requirements evolution" instead of "requirements elicitation"; using keywords which cover only one or two of the three keyword blocks despite being relevant; using only the name of a specific analytics technique (e.g., Long Short-term Memory) and not more general terms included in the identified keywords, e.g., machine learning.

To update the search results, we performed additional search and two-level screening, using the same search query process. The updated search identified 401 after removing duplicates (Table 4). Two-level screening resulted in including 17 additional studies. However, we did not perform a backward and forward reference search during this phase. We also included one study that was not captured by the search query but was recommended by an expert due to its relevance to our research question. We, therefore, selected a total 68 studies to be included in this review.

General Characteristics of the Selected Studies

Of the 68 selected articles, conference proceedings are the most frequent publication type ($n = 41$), followed by journal articles ($n = 16$), workshop papers ($n = 7$), and symposium papers ($n = 4$). All selected studies except one (2009) were published between 2012 and 2020. Figure 2 depicts the total number of the included papers per publication year. Although the number of publications dropped in 2018, in general, there is an increasing trend of publications between 2012 and 2019. For the year 2020, the result is shown as of July 3. A further observation is thus needed to confirm the increasing trend at the end of the year. The median score for study quality was 1 with the interquartile range of 0–1.5 (Appendix 2).

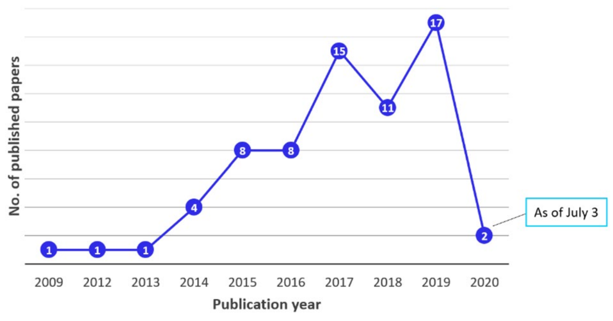


Fig. 2 Publication trend

Types of Dynamic Data Sources Used for Requirements Elicitation

Dynamic Data Sources Used for Automated Requirements Elicitation

Among dynamic data sources, human-sourced data sources have been primarily used as requirements sources. Among the three types of dynamic data sources, the vast majority (93%, $n = 63$) of the studies used human-sourced data sources for eliciting requirements. Only four studies (6%) explored using either machine-generated ($n = 2$) or process-mediated ($n = 2$) data sources. Almost all the studies focused on a single type of dynamic data source. We identified only one study attempting to integrate multiple types of dynamic data sources (1%).

The Specific Types of Dynamic Data Used for Automated Requirements Elicitation

The following seven data sources have been used for automated requirements elicitation: online reviews, micro-blogs, online discussions/forums, software repositories, software/app production descriptions, sensor readings, usage data from system–user interactions, and mailing lists (Table 5). Online reviews are reviews of a product or service that is posted and shown publicly online by people who have purchased a given service or product. Microblogs, which are typically published on social media sites, are a type of blog in which users can post a message in a form of different content formats such as short texts, audio, video, and images. They are designed for quick conversational interactions among users. Online discussions/forums are online discussion sites where people can post messages to exchange knowledge. Software repositories are platforms for sharing software packages or source codes, which primarily contain three elements: a trunk, branches, and tags. This study also

Table 5 Dynamic data used for automated requirements elicitation

Dynamic data	Data descriptions
Online reviews	• Online reviews included app reviews, reviews compiled by experts, and online user reviews. Among the studies which used online reviews, a majority of the studies used app reviews as the sources of potential requirements (75%) [35–54]. Of them, 14 used app reviews from multiple distribution platforms such as Apple AppStore and Google Play to increase the level of generalizability, while eleven used those from a single distribution platform, and one did not specify the number of app distribution platforms • Of the studies which used online reviews, 17% ($n=6$) extracted user reviews of software and video games [55], IoT products [56], compact cameras [57], internet security [58], Jira and Trello [59], and Jingdong.com [60]. Expert reviews were used in the 8% ($n=3$), of which two were from multiple platforms [61, 62], and one was from a single platform [63]
Microblogs	• Microblog data from twitter, Facebook, and Weibo were used for automated requirements elicitation. Of total eleven studies that used twitter, four studies extracted only texts [64–67], while the rest extracted additional metadata [68–74]. The metadata include the number of retweets, likes, lexically similar tweets (i.e., duplicates), twitter followers and friends (i.e., social rank), replies to tweets, as well as hashtags, handles (i.e., indicated by an @ appended with a username), and demographic data of the person who tweeted. • Ali et al. [66] and Han et al. [75] used user comments on Facebook and Weibo, respectively. Seven out of 11 studies that used microdata performed sentiment analysis of tweets [64–66, 68, 69, 72, 73]
Online discussion/forum	Eight studies (12%) elicited requirements from different online discussion forums: online forum posts from the feature tracker of the Password Safe project on SourceForge [76], questions and answers on Stack Overflow [77], and feature requests from open-source forums of SugarCRM, SecondLife, and an Amazon-like portal specifically developed for students [78], OpenOffice online discussion forum [79], the Reddit forum [80–82], and the VLC media player and Firefox web browser forums [83]
Software repositories	Seven studies (10%) leveraged data from software repositories: the Apache OpenOffice issue tracking system [84], issues mined from the Android OS issue tracker [23], the natural language and features from the issue tracking system metadata of the four open-source projects [85], and GitHub [86], GitHub and JIRA issue tracker [87], and a data sink tank containing data from multiple software repositories (e.g., GitHub, JIRA, Jenkins, and SonarQube) [88, 89]
Software/app product descriptions	Five studies (7%) used software product descriptions on Softpedia [90, 91], app change logs [92, 93], or app description page [94]
Sensor readings	• Voet et al. [95] analyzed usage elements, which are the individual activities or parameters that are captured via a handheld grinder, equipped with sensors, for user-centered and data-driven product improvement. Features were extracted and selected from the usage elements and stored in the form of an array of the features per time window (i.e., the coefficients of an autoregressive model of order two). Those selected features were fed into a machine-learning algorithm to predict usage element states. • Liang et al. collected [96] user context data (i.e., time, the location and motion state of the crowd mobile users) and the currently running apps (i.e., mobile applications running at a given time point) via the sensors of the smartphones and used to mine context-aware user requirements. User context data are structured as a 3-tuple, while the currently running apps are represented as the name of the apps. Context-aware user requirements take the form of user behavior patterns, which are represented as frequent item sets of user behavior (i.e., 4-tuples sets of user context and the name of the current running app(s)).
Usage data	• Xie et al. [97] and Yang et al. [98] identified emerging or new user intentions based on users' run-time behavioral patterns and the corresponding environmental context values, when using the Cooperative Research Environment (CoRE) system, an online library system that is modified based on an open-source web application called MyReview. Usage data were structured as a set of feature functions that are defined to reflect the relations between time-series user behavior (i.e., user actions) and the corresponding contextual values and goals • Wüest et al. [99] monitored the system usage to detect requirements violation and observed a sequence of feature usage to better understand user requirements
Mailing lists	Two studies used mailing lists as requirements sources: Apache Common User List [100] and open-source software mailing lists [79]

considered issue-tracking systems as software repositories, which are detailed reports of bugs or complaints written in the form of free texts. Sensor readings are electrical outputs of devices that detect and respond to inputs from a physical phenomenon, which results in a large amount of streaming data. Usage data are run-time data collected when users are interacting with a given system. Mailing lists are a type of electronic discussion forums. E-mail messages sent by specific subscribers are shared by everyone on a mailing list.

Figure 3 depicts the types of dynamic data that have been used for automated requirements elicitation. Online reviews are the most frequently used type of dynamic data for eliciting requirements (53%), followed by micro-blogs (18%) and online discussions/forums (12%), software repositories

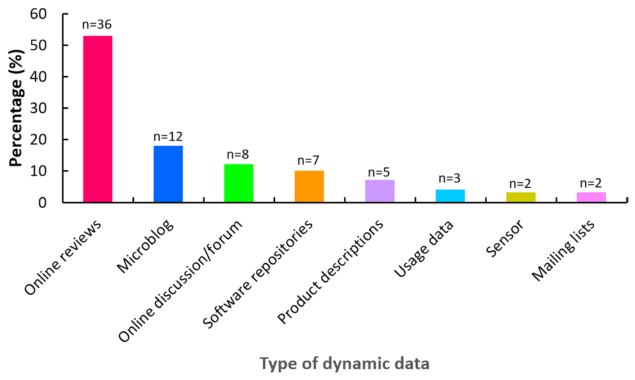


Fig. 3 Types of dynamic data used for automated requirements elicitation

(10%), and software/app product descriptions (7%). Other types of dynamic data include sensor readings (3%), usage data from system–user interactions (4%), and mailing lists (3%).

Several studies used multiple types of human-sourced data to gain complementary information and improve the quality of the analysis. Wang et al. [92] assessed whether the use of app changelogs improves the accuracy of identifying and classifying functional and non-functional requirements from app reviews, compared to the results obtained from the mere use of app reviews. Although there were no additional positive effects of app changelogs on improving the accuracy of automatic requirements classification, their subsequent study [93] shows that the accuracy of classifying requirements in app reviews by augmenting the reviews with the text feature words extracted from app changelogs.

Takahashi et al. in [100] used Apache Commons User List and App Store reviews. However, those two types of datasets were used independently without being integrated to evaluate their proposed elicitation process. Moreover, Stanik et al. [65] used three datasets: app reviews, tweets written in English, and tweets written in Italian. On the other hand, Johann et al. [94] integrated both app reviews and descriptions to provide information on which app features are or are not actually reviewed. In addition, Ali et al. [66] combined tweets for a smartwatch and Facebook comments of wearable and smartwatch.

Some studies used multiple types of software repositories. Morales-Ramirez et al. [84] used two types of datasets obtained from the issue tracking system of the Apache OpenOffice community and the feedback gathering system of SEnerCON, which is an industrial project in the home energy management domain. In their different study [79], open-source software mailing lists, and OpenOffice online discussions were used to identify relevant requirements information. Nyamawe et al. [87] used commits from GitHub repository and feature requests from JIRA issue tracker, while Oriol et al. [89] and Franch et al. [88] considered heterogenous software repositories.

Only one study used multiple types of data sources (e.g., human-sourced data and machine-generated data). Wüest et al. in [99] used both app user feedback (i.e., human-sourced data) and app usage data (i.e., process-mediated data).

Relation of Dynamic Data to an Organization of Interest

The majority of the studies used dynamic data that was external to the organization of interest. Of the 68 studies included in the analysis, 57 studies (85%) used dynamic data which was externally related to a given organization (i.e., data were collected outside of an organization's platforms) [36–77, 80–82, 86–94, 96, 101]. Nine studies (13%) used dynamic data that were collected from platforms belonging to the organization: issue tracking systems [84, 85, 102]; user feedback from the online discussion and open-source software mailing lists [79]; sensors equipped with an intelligent product which is also known as a product embedded information devices (PEID) [95]; software production forum [103]; user feedback tool [99]. On the other hand, only two studies (3%) used both internal and external dynamic data [78, 100].

Additional Use of Domain Knowledge Used for Requirements Elicitation

Only one study considered additional inclusion of domain knowledge in eliciting requirements. Yang et al. [44] combined the app review analysis and the Wizard-of-Oz technique for the requirements elicitation process. The results indicate that integrating the two sources can complement each other to elicit more comprehensive requirements that cannot be obtained from either one of the sources.

Approaches for Automated Requirements Elicitation

Approaches Used for Human-Sourced Data

Since human-sourced data are typically expressed in natural language, natural language processing (NLP) is commonly used for analyzing this type of data. All of the 63 studies which used human-sourced information started the requirements elicitation process by preprocessing the raw data using NLP techniques. Data preprocessing typically involves removing noise (e.g., HTML tags) to retain only text data. Another critical data preparation activity is tokenization, which means splitting the text into sentences and tokens (words, punctuation marks, and digits), respectively.

Further analysis of the text using NLP typically involves syntactic analysis, such as part-of-speech tagging. Two studies have used speech-acts, which are acts performed by a speaker when making an utterance, as parameters to train supervised learning algorithms [79, 84]. For eliciting requirements, nouns, verbs, and adjectives are often identified since they are more likely used for describing requirements-related information than other parts of speech, including adverbs, numbers, and quantifiers [40].

A common preprocessing activity is stopword filtering, which involves removing tokens that are common but carry little meaning, including function words (e.g., “the”, “and”, and “this”), punctuations (e.g., “.”, “?”, and “!”), and special characters (e.g., “#” and “@”), and numbers. Normalization is moreover often carried out by lowercasing (i.e., convert all text data to lowercase), stemming (i.e., reduce inflectional word forms to their root form such as reducing “play”, “playing” and “played” to their common root form of “play”) and lemmatization (i.e., grouping the different inflected forms of words which are syntactically different but semantically equal to be analyzed as a base form, called lemma, such as grouping “sees” and “saw” into a single base form of “see”).

Once the text data have been preprocessed, features are typically extracted for the subsequent modeling phase. Feature extraction can be done using a bag of words (i.e., simply count occurrences of tokens without considering word order nor normalizing counters), *n*-grams (i.e., extract the contiguous sequence of *n* tokens such as bi-gram which indicates the extraction of token pairs), and collocations (i.e., extract a sequence of words that co-occur more often than by chance, for example, “strong tea”). To evaluate how important a word is for a given document, a bag of words are often weighted, using a weighting scheme such as term frequency-inverse document frequency (tf-idf), which gives high weights to words that have a high frequency in a particular document, while having a low frequency in an entire set of documents. Other common features are based on syntactic or semantic analysis of the text (e.g., part-of-speech tags). Sentiment analysis, which is the automated process of identifying and quantifying the opinion or emotional tone of a piece of text through NLP, was used in 18 studies (38%), either to feed into algorithms as features to increase the accuracy of the algorithms or to understand user satisfaction.

After preprocessing the human-sourced data and extracting features for data modeling, the next step of requirements elicitation was to perform either classification or clustering. Classification refers to classifying (text) data into predefined categories related to requirements, for example, classifying app reviews into bug reports, feature requests, user experiences, and text ratings [38]. Classification has been performed using three approaches: machine learning (ML), rule-based classification, or model-oriented approaches. In the ML approach, classification is performed by a model built by a learning algorithm based on pre-labeled data.

In the ML approach, various learning algorithms automatically learn statistical patterns within a set of training data, such that a predictive model is able to predict a class for unseen data. In most studies, ML relied on supervised ML. In supervised ML, a predictive model is built based on instances that were pre-assigned with known class labels (i.e., training set). The model is then used to predict a label associated with unseen instances (i.e., test set). A downside with supervised ML is that it typically requires a large amount of labeled data (i.e., ground-truth set) to learn accurate predictive models.

To reduce the cost of labeling a large amount of data, a few studies used the active learning paradigm and semi-supervised machine learning for classification. Active learning enables machines to wisely select unlabeled data points to be labeled next in a way that optimizes a decision boundary created by a given learning algorithm and interactively queries the user to label the selected data points to improve classification accuracy. Semi-supervised learning is an intermediate technique between supervised and unsupervised ML, which utilizes both labeled and unlabeled data in the training process.

Rule-based classification is a classification scheme that uses certain rules, such as language patterns. Rule-based classification excels in performing simpler tasks where domain experts can define rules, while classification using ML works well for the tasks which are easily performed by humans but where (classification) rules are hard to formulate. However, listing all the rules can be tedious and needs to be hand-crafted by skilled experts with abundant domain knowledge. Moreover, rules might need to be refined as new datasets become available, which requires additional resources and limits scalability [77]. A model-oriented approach, which includes utilizing conceptual or meta-models, are applied to define and relate the mined terms and drive classification.

On the other hand, clustering has been performed using either topic modeling or more traditional clustering techniques. Topic modeling is an unsupervised (i.e., learn from unlabeled instances) dimension reduction and clustering technique, which aims to discover hidden semantic patterns in the collection of a document. Topic modeling is used to represent an extensive collection of documents as abstract topics consisting of a set of keywords. In automated requirements elicitation, topic modeling is mainly used for either discovering system features or grouping similar fine-grained features that are extracted using different approaches into high-level features. Traditional clustering is an unsupervised ML technique that aims to discover the intrinsic structure of the data by partitioning a set of data into groups based on their similarity and dissimilarity. Among the selected studies, traditional clustering has been mainly used to discover inherent groupings of features in requirements-related information.

Some studies have performed clustering after classification. Classification was first performed to identify and classify requirements-related information, using machine learning or rule-based classification. Clustering is then applied to the identified requirements-related information (e.g., improvement requests), while ignoring data irrelevant to requirements, to discover inherent groupings of features, using topic modeling or traditional clustering. Table 6 provides a more detailed description of the automated approaches proposed in each study.

Figure 4 depicts the descriptive statistics of the approaches for automated requirements elicitation used in the selected studies. For classification, the most commonly used approach was based on the ML approach (60%),

Table 6 Summary of the automated requirements elicitation approaches for human-sourced data that is grouped by classification approach and clustering approach

Classification approach	How
Machine learning	• Classification with supervised learning has been performed in numerous studies [36, 38, 39, 42, 45, 47, 48, 50, 51, 53, 55, 56, 58–60, 64–71, 77, 79–82, 84, 85, 87, 91–93, 103, 104] to classify textual data into pre-defined labels that are relevant for performing requirements elicitation, including identification of relevant requirements sources and stakeholders as well as extraction of candidate features. • Frequently used learning algorithms were Naïve Bayes, multinomial Naïve Bayes, Support Vector Machine, and logistic regression. Decision tree and random forest were also used in some studies [45, 69, 85]. Several different algorithms are often applied in the same study to compare the performance of classification. • Active learning based on uncertain sampling, which selects data points which a model is most uncertain about for manual labeling, was applied to classify app reviews into a feature request, bug report, rating, and user experience [35]. • A semi-supervised learning technique was used to classify app reviews into either functional or non-functional requirements [43].
Rule-based classification	• Rule-based classification has been applied to identify and classify relevant requirements information, to identify software features [40, 41, 46, 61, 62], and to elicit requirements [94]. • Rule-based classification has been performed: using collocation finding algorithms [40, 46]; grammar-based and delimiter-based strategies [76]; phrase search [86]; conducting language pattern matching which results are post-processed with Part-of-Speech tagging [57]; n-gram analysis [102]. It has also been performed based on language patterns [37, 61], or linguistic tags [62], propagation rules based on syntactic relations [41], or the SAFE patterns which consist of Part-of-Speech patterns and sentence patterns [94].
Model-oriented	• A user-oriented conceptual modeling approach was used for conceptualizing potential requirements in the form of consumer preferences and classifying and ranking the preferences to identify software features [73]. Highly-ranked consumer preferences in the conceptual model were further transformed into a i* goal model to explicate early/high-level system requirements [72]. • Q-rapids, which is a quality-aware agile software development framework, was used to perform data-driven elicitation of quality requirements [88, 89]. The data collected from heterogeneous software repositories are fed into a quality model to compute the quality of the software. If the quality level violates the user-defined threshold, quality alert is automatically created. When the alert is raised, candidate quality requirements (QRs) are identified, using the QR pattern catalog. The catalog consists of a set of QR patterns, which are bound to the quality model for matching the appropriate candidate QR patterns with the raised quality alert. The candidate QR patterns are then presented to stakeholders to elicit QRs.
Clustering approach	**How**
Topic modeling	• Within requirements engineering, different topic modeling algorithms have been used, depending on the length of the data. Latent Dirichlet Allocation (LDA), which is a common topic modeling algorithm, has been applied to app reviews in the studies [40, 44, 46, 100]. Higashi et al. [54] improve LDA by keyword expansion. Zhao and Zhao [47] used hierarchical LDA to extract software features. However, LDA is not suitable for analyzing shorter texts such as tweets due to the sparsity of word co-occurrence patterns in the individual document • To overcome the problem, the Biterm Topic Model (BTM) has been developed for short texts [105]. In one study [68], BTM was applied to tweets to discover topics that are related to users' experiences.
Traditional clustering	• K-means clustering has been used to extract software features from online reviews [41, 58, 63, 75, 78, 90], and to cluster informative app reviews [53] Jiang et al. [41] used S-GN to cluster online reviews, while Cleland-Huang et al. in [78] used the modified Spherical K-Means to extract and cluster feature requests from threads in open discussion forums. • Kang et al. [91] used the bagging clustering algorithm, which combines the EM, K-means, and MTree clustering algorithms for grouping similar data to select transfer instances. The transfer instances are further utilized to build classifier models.
Rule-based clustering	One study [74] proposed an unsupervised clustering algorithm, which extracts basic context items from user feedback (e.g., the affected platform, device, app- and system version), using pre-defined keyword lists, word vector representations, and text patterns.

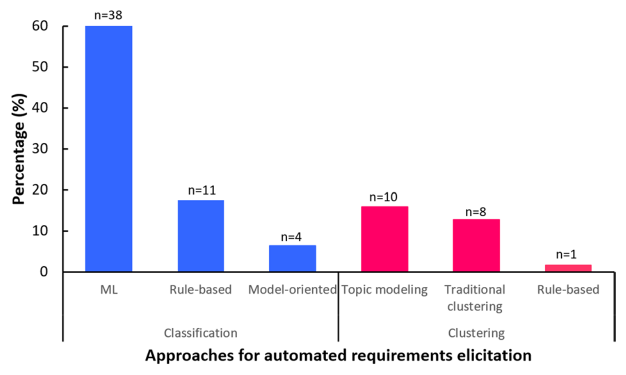


Fig. 4 Techniques used for requirements elicitation from human-sourced data that are grouped according to classification (i.e., machine learning (ML), rule-based classification, and model-oriented approach) and clustering (i.e., topic modeling, traditional clustering, and rule-based unsupervised NLP)

followed by rule-based classification (17%) and model-oriented approach (6%). For clustering, topic modeling (16%) was the most commonly used approach, followed by more traditional clustering techniques (13%) and unsupervised rule-based clustering (2%).

In nine studies, two different approaches have been combined. Two studies performed classification with supervised ML for filtering and subsequently conducted clustering with topic modeling [47, 68]. Guzman et al. [68] first ran Multinomial Naïve Bayes and Random Forest, which are both supervised learning algorithms, to extract tweets that request software improvement. Biterm Topic Model, which is a topic modeling used for short texts, was then used to group semantically similar tweets for software evolution. Zhao and Zhao [47] ran a supervised deep-learning neural network was first used to extract software features, and their corresponding sentiments and hierarchical LDA was subsequently to extract hierarchical software features with positive and negative sentiments.

Two studies performed classification using ML, which was followed by unsupervised clustering analysis [53, 58]. Jiang et al. [58] used Support Vector Machine, or a supervised machine-learning algorithm, for pruning incorrect software features that were extracted from online reviews. K-means clustering, an unsupervised clustering analysis, was then performed to categorize the extracted features into semantically similar system aspects. Sun and Peng [53] first used Naïve Bayes, a supervised machine-learning algorithm, for filtering informative comments, which were subsequently clustered using K-means, an unsupervised clustering analysis.

Jiang et al. [41] first performed rule-based classification based on syntactic parsing and sentiment analysis to extract opinions about software features and their corresponding sentiment words. Subsequently, S-GN, whose base algorithms are a type of K-means clustering, was performed to cluster similar opinion expressions about software features into a category which represents an overall, functional, or quality requirements. On the other hand, Bakar et al. [63] combined unsupervised clustering analysis and topic modeling in which K-means was first run to identify the similar documents. They then performed latent semantic analysis, which is a type of topic modeling, to group similar software features within the documents.

Guzman and Maalej [40] and Dalpiaz and Parente [46] first extracted software features based on rule-based classification, which uses collocation finding algorithm and the LDA was subsequently applied to group similar software features. Zhang et al. [60] first used linear regressions based on supervised ML to select helpful online reviews. Then conjoint analysis (i.e., a statistical technique used in market research to assess and quantify the consumers' values on product features or service) was performed to assess the impact of the features from helpful online reviews on the consumers' overall rating.

In several studies, visualization has been provided to help requirements engineers efficiently sift through and effectively interpret the most important requirements-related information. Bakiu and Guzman [55] first performed the aggregation of features. The results were then visualized at two levels of granularity (i.e., high-level and detailed). Sun and Peng [53] first extracted scenario information of similar user comments and then aggregated and visualized as aggregated scenario models. Software features [52] and technically informative information from the potential requirements sources [64, 86] were summarized, ranked, and visualized using word clouds. Luiz et al. [49] summarized overall user evaluation of the mobile applications, their features, and the corresponding user sentiment polarity and scores in a single graphical interface. Oriol et al. [89] implemented a quality-aware strategic dashboard, which has various functionalities (e.g., quality assessment, forecasting techniques, and what-if analysis) and allows for maintaining traceability of quality requirements generation and documentation process. Wüest et al. [99] fused user feedback and correlated GPS data and visualize the fused data on a map, equipping the parking app with context-awareness.

Techniques Used for Process-Mediated Data

The two studies that used process-mediated data focused on eliciting emerging requirements through observations and analysis of time-series user behavior (i.e., run-time observation of system–user interactions) and the corresponding environmental context values [97, 98]. In both studies, Conditional Random Fields (CRF), which is a statistical modeling method, was used to infer goals (i.e., high-level requirements).

Xie et al. [97] proposed a method to elicit requirements consisting of the three steps. First, a computational model is trained and built based on pre-defined user's goals in the domain knowledge, using supervised CRF to infer user's implicit goals (i.e., outputs) from the observation and analysis of run-time user behavior and the corresponding environmental values (i.e., inputs). After the goal inference, the user's intention (i.e., the execution path) for achieving a given goal is obtained by connecting the situation (i.e., a time-stamped sequence of user behavior that is labeled with a goal and environmental context values) labeled with the same goal into a sequence. Finally, an emerging intention, which is a new sequence pattern of user behavior that has not been pre-defined or captured in the domain knowledge base, is detected.

An emerging intention can occur in three cases; when a user has a new goal; when a user has a new strategy for achieving an existing goal; when a user cannot perform operations in an intended way due to system flaws. Requirements, thus, can be elicited by validating emerging intentions by domain experts based on the analyses of goal transition, divergent behaviors from the optimal usage, and erroneous behavior.

In the analysis of goal transition, domain experts look at two goals that frequently appear consecutively based on the results of goal inference with a high confidence level assigned by the CRF and elicit requirements that make the goal transition smoother.

In the analysis of divergent behavior, domain experts focus on user behaviors that deviate from an expected way to operate the system because the user's irregular behavior may indicate user's misunderstanding of required operational procedures, dissatisfaction with the system, and emerging desires. Those divergent behaviors are given a low confidence level by the CRF model.

In the analysis of erroneous behavior, requirements can be elicited by investigating the error reports with high occurrences that may reflect users' emerging desires that are not supported by the current system. In addition, requirements can be elicited from user behaviors, which are actually normal behavior but are mistakenly considered as erroneous due to the system flaws. The proposed method is assumed to be used in a sensor-laden computer application domain. Thus, it may also be applicable to machine-generated data. The main challenge, however, is to increase the level of automation for analyzing potential emerging intentions and users' emerging requirements.

Yang et al. in [98] used CRF to infer goals based on a time-stamped sequence of user behavior that is labeled with a goal and environmental context values, which is called a situation. Based on the results of goal inference, intention inference was performed by relating a sequence of situations that are labeled as the same goal. When an intention has not been pre-defined in the domain knowledge base, the intention is detected as an emerging intention and exported as possible new requirements for future system development or evolution.

However, the method proposed in both studies still requires a substantial degree of human oracles, which needs to be reduced in future research to increase the scalability and promote the implementation of their approach in real-life settings. In addition, the proposed method does not yet support diverse requirements. The method proposed by Xie et al. [97], capture only emerging functional but not non-functional requirements. The approach proposed in [98] can only support the identification of the low-level design alternatives (i.e., new ways of fulfilling a given intention).

Notably, Wüest et al. in [99] proposed to use both human-sourced and process-generated data. Their approach is based on the control loop for self-adaptive systems for collecting and analyzing user feedback (i.e., human source data) as well as system usage and the location data (i.e., GPS data). The analysis is driven by rules or models of expected system usage. The system decides how to interpret the results of the analysis and modify its behavior at run-time, which allows for understanding changing user requirements for software evolution.

Techniques Used for Machine-Generated Data

Voet et al. [95] first extracted goal-relevant usage elements as features, from the data recorded via a handheld grinder, a type of product embedded information devices (PEID) equipped with sensors and onboard capabilities. Feature selection was then performed to reduce system workload and improve the prediction accuracy of the machine-learning algorithm, compared to using raw sensor data. Specifically, the support vector machine classifier, which is a supervised machine-learning algorithm, was used to build and train the model to predict the four different usage element states. The model was then tested on the sensor data from the two different usage scenarios that have not been used for training. The collection of the predicted usage element states, or user profiles, can be analyzed manually or by clustering to identify the deviation from the intended optimal usage profile. Requirements can be inferred by analyzing users' deviant behaviors.

Liang et al. [96] mined user behavior patterns from instances of user behavior, which consist of user context (i.e., time, the location and the motion state of the crowd mobile users) and the currently running apps, using Apriori-M algorithm, which is an efficient algorithm based on Apriori algorithm that is used for frequent item set mining. User behavior patterns, which infer emergent requirements or requirements changes, are ranked and used for service recommendation. Service recommendation is performed periodically, using the service recommendation algorithm. The algorithm takes mined user behavior as inputs and outputs the apps to remind the user. In service recommendation, matching is performed between the current user context and the context of user behavior patterns mined from mobile crowd users, according to the ranking order. If the two matches, the mobile app(s) in the user behavior patterns are automatically recommended to the user as solutions to meet the requirements inferred from user behavior patterns.

In summary, most of the existing solutions support the elicitation of requirements from a single data source, primarily from human source data. There is a lack of methods to support requirements elicitation from heterogeneous data sources. In addition, only a few studies have supported context-awareness and real-time data processing and analysis. Those features are crucial to enable continuous and dynamic elicitation of requirements, which are especially important for context-aware applications and time-critical systems such as health systems. Moreover, many studies lack the argument on how each proposed solution help processing a large volume of data.

Evaluation Methods

Evaluation methods include three components: evaluation approach, concept, and metrics. Of the 68 selected studies, controlled experiments were the most frequently applied approach for evaluating the proposed artifact (75%), followed by a case study (19%) and a proof of concept (6%) (Fig. 5a).

Among the 51 studies that used controlled experiments, 46 studies compared the results produced by the proposed artifacts against a manually annotated ground-truth set. For example, Bakiu and Guzman [55] compared the performance of multi-label classification against a manually created golden standard in classifying features extracted from unseen user reviews into different dimensions of usability and user experience.

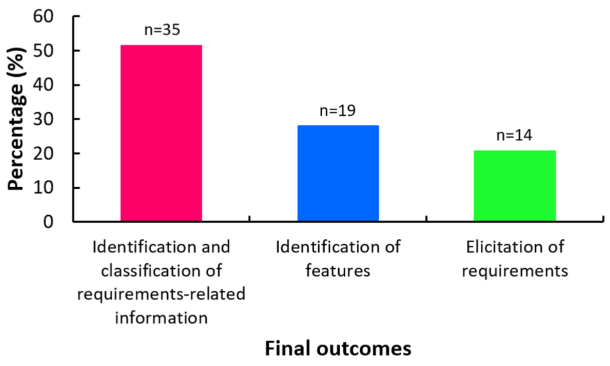


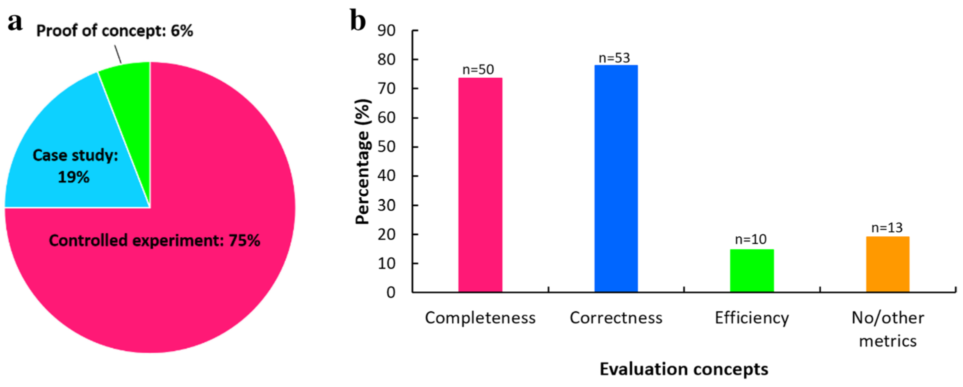
Fig. 6 Final outcomes of automated requirements elicitation

Only three studies compared the performance of the proposed artifact with the results of manual analysis without the aid of automation [57, 62, 78]. For example, Bakar et al. [62] compared the software features that were extracted using their proposed semi-automated method with those that were obtained manually.

Two studies conducted an experiment in different ways. Liang et al. [96] used a longitudinal approach for conducting an experiment. They compared obtained user behavior patterns with those that were collected after some time interval to confirm the correctness of the Apriori-M algorithm. Abad et al. [44] compared Wizard-of-Oz (WOz) and user review analysis qualitatively. In a few studies [46, 88, 90], the proposed techniques have been evaluated with intended users. The rest of the studies used a case study or a proof of concept as an evaluation approach.

Most frequently used evaluation concept was correctness (78%), followed by completeness (74%), no/other metrics (13%), and efficiency (10%) (Fig. 5b). Other metrics, for example, include usability, creativity, the intended user's perceived usefulness, and satisfaction. Most of the studies combined several evaluation concepts. Three different combinations of the concepts were identified: (1) completeness and correctness ($n = 42$), (2) completeness and correctness and efficiency ($n = 7$), and (3) correctness and efficiency

Fig. 5 **a** Evaluation approach, **b** Evaluation concepts



($n = 2$). In most cases, the correctness and completeness were assessed using precision (i.e., the fraction of correctly predicted instances among the total predicted instances) and recall (i.e., the fraction of correctly predicted positive instances among all the instances in actual class), respectively. In addition, F-measure was also used to address a trade-off between precision and recall.

Efficiency has been assessed in terms of the size of training data [38, 39], the time to recognize and classify software features [76], the time required to identify relevant requirements information for both manual and automated analysis [57], the time taken to complete the extraction of software features [63], the time and space needed to build the classification model [48, 50, 51], and the total execution time of the machine-learning algorithm [78]. The user's perceived efficiency was measured using a 5-point Likert scale [89]. (Fig. 6)

The Outcomes of the Automated Requirements Elicitation

Expression of Final Outcomes Produced by the Automated Part of Requirements Elicitation

Outcomes of the automated requirements elicitation have been classified into the following three categories: (1) identification and classification of requirements-related information, (2) identification of candidate features related to requirements, and (3) elicitation of requirements (Table 7). Only 21% of the studies have enabled the automated elicitation of requirements. A majority of the studies have resulted in automated identification and classification of requirements-related information (51%), or identification of candidate features related to requirements (28%) (Fig. 5).

Identification and classification of requirements-related information have been made by classifying dynamic data into different classes of issues based on; relevance to different stakeholders for identifying responsibilities, the technical relevance for filtering only relevant data (e.g., classifying into either feature request or other), and types of technical issues to be inspected (e.g., classifying into feature requests,

Table 7 Expression of final outcomes

Final outcomes	Description of the final outcomes
Identification and classification of requirements-related information	• Although Portugal et al. [86] identified only the sources of requirements-related information, many studies further classify requirements-related information based on stakeholders, technical relevance, and types of requirements issues to be inspected such as feature requests and bug reports. • A multi-label classification technique has been used in the studies [69, 70] to classify tweets into three high-level stakeholder groups (i.e., technical, non-technical, and general public) based on the relevance to each stakeholder group. • In the studies [59, 71, 76, 77, 84], classification has been used to filter only system relevant data, which contributes to the significant reduction of data to go through. Filtering has been done, using supervised machine-learning algorithms [59, 71, 77, 84] and rule-based classification [76]. • More detailed classification than filtering has been performed in the studies [35, 37–39, 42, 43, 45, 50, 51, 54–57, 64–66, 79–82, 87, 92, 93, 102–104, 106], for example, by classifying app reviews into bug reports, feature requests, user experiences, or text ratings [38], by classifying user reviews into specific dimensions of usability and user experiences [55], and by classifying app reviews into functional and non-functional requirements [43]. • Classification of requirements-related information has been done using machine-learning approach [35, 38, 39, 42, 43, 45, 50, 51, 55, 56, 64, 79, 92], rule-based classification [37, 57, 102], and topic modeling [54]
Identification of candidate features related to requirements	• Identification of candidate features has been done, using rule-based classification [40, 46, 61, 62, 100], topic modeling [49, 52], traditional clustering (i.e., K-means [75] and the BIRCH algorithm [90]), a conceptual model [72, 73], and rule-based clustering [74]. • In the several studies, classification was first performed, using supervised machine learning [47, 48, 58, 60, 68] or ruled-based classification [41, 46] before identifying candidate software features using topic modeling [46, 47, 68], traditional clustering [41, 58], word clouds [48], or conjoint analysis [60]. • In contrary, Kang et al. [91] used instance-transfer learning method. The method first performed the bagging clustering algorithm to select instances, whose results are used for building a RNN model to predict missing features.
Elicitation of requirements	• Requirements have been elicited at an abstract level in the form of textual descriptions [36, 44, 78, 94], goal models [72], and aggregated scenarios [53]. • In a few studies, requirements have been elicited at a lower level than the above studies. Liang et al. [96] inferred emerging requirements and requirements changes from mined user behavior patterns of crowd mobile users • Xie et al. [97] and Yang et al. [98] elicited requirements in the form of emerging from users' run-time behavioral patterns, while Voet et al. identified potential design changes from the usage profile that is mined from sensor data [95]

bug reports, user experiences, and user ratings, and classifying into functional or non-functional requirements). Some studies performed classification at a deeper level (e.g., classifying into four types of non-functional requirements (i.e., usability, reliability portability, or performance, or functional requirements).

Identification of candidate features related to requirements refers to discovering functional components of a system. Features, however, typically have less granularity than requirements and do not tell what behavior, conditions, and details would be needed to obtain the full functionality. They, thus, need to be further processed to become full requirements.

Elicitation of requirements has been done mostly at high level. Most of them elicited requirements at high level in the form of goals, aggregated scenarios, or high-level textual requirements. Franch et al. [88] and Oriol et al. [89] semi-automated the elicitation of complete requirements in the form of user stories and requirements specified in semi-formal language.

Degree of Intended Automation

A proposed artifact was classified into the two levels of the intended automation: intended full automation or semi-automation. Of note is that we consider artifacts that support the automation of requirements elicitation either entirely or partially. Artifacts are classified into intended full automation in the following two circumstances: (1) when the proposed part is automated without human intervention for completion or (2) when only minimum interactions are needed for completion. Minimum human interaction is defined as human oracles being in the loop once at the initial stage of the elicitation process, which includes the creation of the ground-truth set and conceptual models as well as the specification of a set of keywords and language patterns. Based on the definitions, the majority of the proposed methods (84%) were intended to be fully automated, while the rest are semi-automated methods that require human oracles to be in the loop for each iteration of the process.

Additional Requirements Engineering Activity Supported Through Automation

The majority of the selected studies exclusively focused on enabling requirements elicitation from dynamic data, without considering other requirements engineering activities. Of the 68 studies included in the analysis, 50 studies (74%) exclusively proposed methods to enable automated requirements elicitation, while 18 studies (26%) supported other requirements engineering activities in addition to requirements elicitation. Prioritization was the most frequently supported additional requirements engineering activity ($n = 11$), followed by elicitation for change management ($n = 7$), and documentation ($n = 2$). More detailed information is provided in Table 8.

Table 8 Additional requirements engineering activity supported through automation

Requirements engineering activity	Description of activity supported through automation
Elicitation	The majority of the studies have exclusively focused on requirements elicitation [35–40, 42–47, 50, 51, 53–57, 59, 61–67, 69–71, 74–80, 82, 84, 86, 90–96, 100, 103, 104].
Elicitation plus additional activities	• Prioritization of the elicited requirements refers to ranking the elicited requirements according to a certain criterion to promote effective resource allocation. Prioritization was supported in the studies [41, 48, 49, 52, 58, 60, 68, 72, 73, 81, 87–89]. • Elicitation for change management is to elicit emerging requirements for software evolution, which has been supported in the studies [52, 85, 87, 97–99, 102]. • Documentation of quality requirements has been enabled by two studies [88, 89], by automatically specifying user stories or semi-formally written requirements in a product backlog, respectively.

Discussion

We conducted a systematic literature review on the existing data-driven methods for automated requirements elicitation. The main motivations for this review were two-fold: (1) using dynamic data has the potential to enrich stakeholder-driven requirements elicitation by eliciting new requirements which cannot be obtained from other sources, and (2) no systematic review has been conducted on the state-of-the-art methods to elicit requirements from dynamic data from unintended digital sources. Of 1848 records retrieved from 6 electronic database search and 1017 articles identified through backward and forward reference search, we selected 51 studies that met our inclusion criteria and included in the final analysis to answer the following three research questions. RQ1: What types of dynamic data are used for automated requirements elicitation? RQ2: What types of techniques and technologies are used for automating requirements elicitation? RQ3: What are the outcomes of automated requirements elicitation? In the following sections, we provide a discussion of the main findings, the identified research gaps, and issues to be addressed in future research.

RQ1: What Types of Dynamic Data Are Used for Automated Requirements Elicitation?

Existing research on data-driven requirements elicitation from dynamic data sources has primarily focused on utilizing human-sourced data in the form of online reviews, micro-blogs, online discussions/forums, software repositories, and mailing lists. The use of online reviews was substantially more prevalent, compared to other types of human-sourced data. The result indicates the current data-driven requirements elicitation is largely crowd-based. On the contrary, process-mediated and machine-generated data sources have only, in some instances, been explored as potential sources of requirements. The predominance of human-sourced information is rather expected and can be explained by the following two reasons: (1) users' preferences and needs regarding system are typically explicitly expressed in natural language, from which it is—relatively speaking—straightforward to obtain requirements compared to process-mediated and machine-generated data, and (2) there are abundant sources of human-sourced data that are publicly available and readily accessible.

Much more research is, thus, needed to develop methods capable of eliciting requirements from process-mediated and machine-generated data that are not expressed in natural language and from which requirements need to be inferred. There is still a lack of methods to infer requirements as well as evidence regarding the applicability of the proposed approach to more diverse types of process-mediated and machine-generated data. Process-mediated and machine-generated data enable run-time requirements elicitation [19]. They also help system developers to understand usage data and the corresponding context, which allows elicitation of performance-related as well as context-dependent requirements [19]. In addition, almost all of the studies have focused on using only a single type of dynamic data and typically also a single data source.

A few studies have utilized multiple human-sourced data sources; however, there has been only one attempt to combine different types of dynamic data sources. As such, there is currently insufficient evidence that using multiple types of data leads to more effective requirements elicitation, but it remains an open issue that merits investigation. We believe that research in this direction would be highly interesting in an attempt to improve data-driven requirements elicitation, both in terms of the coverage and quality of the elicited requirements. Utilizing semantic technologies can be useful for enabling the integration of heterogeneous data sources [107].

In addition, only one study integrated dynamic data and domain knowledge to elicit requirements [44]. The results from that study indicate the potential benefits of using dynamic data together with domain knowledge to elicit requirements that cannot be captured using either one of the data sources. It is likely that domain knowledge, which is typically relatively static but of high quality, can help to enrich data-driven requirements elicitation efforts from dynamic data sources. A larger number of studies are needed to confirm the impacts of integrating domain knowledge with dynamic data on the quality and diversity of outcomes obtained from the automated requirements process.

RQ2: What Types of Techniques and Technologies Are Used for Automating Requirements Elicitation?

Techniques Used for the Automated Requirement Elicitation

Human-sourced data are typically expressed in natural language, which is inherently difficult to analyze computationally due to its ambiguous nature and lack of rigid structure. In all the selected studies, human-sourced data have been (pre-)processed using natural language processing techniques to facilitate subsequent analysis. Although the techniques used for preprocessing varies across studies, data cleaning, text normalization, and feature extraction for data modeling are frequently performed preprocessing steps in automated requirements engineering. Commonly used features include surface-level tokens, words, and phrases, but also syntactic (e.g., part of speech tags) and semantic features (e.g., the positive/negative/neutral sentiment of a sentence). After data preparation and feature extraction, data modeling or analysis for the purpose of requirements elicitation is typically performed using classification or clustering, or classification followed by clustering.

Classification in the context of automated requirements elicitation involves either of the following three tasks: (1) filtering out data irrelevant to requirements, (2) classifying text based on the relevance to different stakeholder groups, or (3) classifying text into different categories of technical issues, such as bug reports and feature requests. The classification tasks have been tackled using either rule-based approaches or machine learning, which is mostly done within the supervised learning paradigm. Although supervised machine learning can achieve high predictive performance in a well-defined classification task, it requires access to a sufficient amount of human-annotated data. As a result, many studies involved human to annotate data into pre-defined classes. The labeling task, however, is labor-intensive, time-consuming, and error-prone due to a considerable amount of noise and the ambiguous nature inherent in natural language [35].

Two solutions have been proposed to reduce the cost of labeling a large amount of data: active learning [35] and semi-supervised machine learning [43]. Dhinakaran et al. in [35] showed that classifiers trained with active learning strategies outperformed in classifying app reviews into feature requests, bug reports, user rating, or user experience than the baseline classifies that were passively trained on the randomly selected dataset. Deocadez et al. in [43] demonstrated that three semi-supervised algorithms (i.e., Self-training, RASCO, and Rel-RASCO) with four base classifiers achieved comparable predictive performance as that of classical supervised machine learning in classifying app reviews into functional or non-functional requirements. Although there is not a sufficient number of studies to draw a generalizable conclusion, classification using active learning and semi-supervised machine-learning strategies may have similar potential as conventional supervised machine learning in identifying and classifying requirements-related information, but requires a much smaller amount of labeled data compared to conventional supervised machine learning.

Another issue that needs to be addressed when using supervised learning is that human-sourced data sources include a significant proportion of non-informative and irrelevant data. Eliciting requirements from this source is thus often compared to "looking for a needle in a haystack" [70]. This leads to a highly unbalanced class distribution in terms of the non-informative and irrelevant data compared to the informative and relevant classes. The underlying class distribution largely affects the performances of machine learning-based classifiers [42, 71]. In one study [42], the precision, recall, and F_1 measures for the under-represented classes were worse than those for the better-represented classes. Given that the classes relevant to requirements are not represented equally in most real-life occasions, the issue needs to be addressed in future research. One possible solution to resolve this issue may be applying different sampling techniques such as Synthetic Minority Oversampling Technique (SMOTE) to the training set to increase the number of instances in the class with fewer observations [71, 84].

Contextualization may be another possible solution, which is done by filtering out non-informative and irrelevant data. Several studies [47, 53, 58, 68] have used supervised classification before performing finer-grained classification or clustering. Filtering out noisy data can improve the classification or clustering accuracy. It also helps requirements engineers pinpoint the data relevant to requirements by automatically discarding non-informative data for requirements elicitation [69] as well as supports efficiently distributing data to the appropriate stakeholders within an organization [69]. Since contextualization can reduce the volume of data to be processed further, it mitigates the volume issue of Big Data.

Various supervised learning algorithms have been used to automate the requirements elicitation process. However, there is no "one-size-fits-all" algorithm that performs best for every single case, which is often referred to as the "No free lunch" theorem [108]. Experimenting and comparing many different algorithms for a specific problem demands time and domain knowledge related to machine learning from requirements engineers in addition to routine work. It would thus be helpful for them if the support tool were to accommodate functions that automatically identify and recommend the best algorithm among possible options.

Moreover, it would be even more valuable if the tool supports automatic optimization of the parameter configuration, which includes preprocessing, selection of machine-learning features, hyper-parameter settings, and evaluation metrics. Supervised machine learning has mainly been used for identifying and classifying data into pre-defined categories related to requirements. This is because supervised machine learning works well for tasks for which classification rules are difficult to formulate. Nevertheless, it requires a sufficient amount of human-annotated data to build a reliable predictive model, which is a time-consuming and error-prone task. On the other hand, rule-based classification, which was the second most frequently used classification approach, excels in performing simpler tasks for which rules can be formulated. In the literature, rule-based classification has been used for identifying candidate features more frequently than identifying and classifying requirements-related information. For rule-based classification to function well, however, sound domain knowledge is required to appropriately define rules that drive the classification process and determine the effectiveness of the classification.

Clustering has been used primarily for identifying candidate features or grouping semantically similar features. In the selected studies, clustering has been performed, using topic modeling or traditional clustering, which can be valuable alternatives to supervised learning in the absence of labeled historical data. More than half the studies that used clustering first classified data into pre-assigned categories relevant to requirements, which was primarily done using supervised machine learning or rule-based classification. Clustering is subsequently performed on the requirements-related information identified by classification, using topic modeling or traditional clustering. Those unsupervised machine-learning techniques, however, often lead to less accurate results than supervised leaning since there is no knowledge about output data.

The effectiveness of clustering can be affected by many factors (e.g., the number of clusters and selection of initial seeds), and evaluating unsupervised learning is problematic due to a lack of well-defined metrics. This may be a reason that classification is performed before clustering. Nevertheless, there are some efforts to ensure high quality of clustering. Cleland-Huang et al. [78] proposed the automated forum management (AFM) system that employs Stable Spherical K-Means (SPK) to mine feature requests from discussion threads in open source forums. In their study, Normalized Mutual Information (NMI) was computed to evaluate and ensure the quality of the cluster. In addition, since the selection of initial seeds highly influence on clustering results, the problem is mitigated by applying consensus clustering for the initial clustering. On the other hand, Sun and Peng [53] used the recommended cluster number (RCN) to determine the optimal number of clusters. There are also other metrics available to evaluate the quality of clustering, such as the Silhouette index. However, the consensus has not been reached regarding which measure to use for the evaluation because it depends on the nature of data and the desired clustering task.

Moreover, only a small proportion of the studies supported the visualization of the obtained results. Data visualization increases the interpretability of the results by leveraging visual capacity, which helps identify new and hidden patterns, outliers, and trends [16]. It also facilitates communication among different stakeholders within an organization. Providing visualizations, thus, is recommended to help requirements engineers understand the results and make a subsequent decision more efficiently and effectively.

Process-Mediated and Machine-Generated Data Sources

As described in the previous section (i.e., “RQ1: What types of dynamic data are used for automated requirements elicitation?”), our results indicate that there is a huge research gap in eliciting requirements from process-mediated and machine-generated data. Much more research should focus on exploring the methods to elicit requirements from data that are not written in natural language. Only two studies leveraged process-mediated data, both utilizing CRF, to infer goalswhich are high-level requirements. More research is need to develop methods and algorithms to elicit requirements from various types of process-mediated data.

Likewise, machine-generated data were used as requirements sources in two studies. Liang et al. [96] proposed to use the Apriori-M algorithm to infer context-aware requirements from behavior patterns that are mined from the runtime behavior of the mobile user. The results of the analysis lead to provide the user solutions that satisfy the inferred requirements. On the other hand, Voet et al. [95] proposed a method to classify goal-relevant usage element states using supervised machine learning and infer requirements based on the deviation from the optimal usage profile, which can be detected by manual analysis or unsupervised clustering.

Given that IoT data are one of the main driving forces of Big Data generation, there is a pressing need to develop a framework to elicit requirements from IoT data. Applying semantic technologies may be a promising solution to help machines interpret the meaning of data by semantically representing raw data in a human/machine interpretable form [107], which can facilitate the automatic requirements elicitation from large volumes of heterogeneous IoT data.

Evaluation Methods

Rigorous evaluation is essential for ensuring that a proposed artifact meets its intended objectives, justifying its effectiveness and/or efficiency, and identifying its weaknesses, which need to be rectified in future work. The artifacts proposed in most of the identified studies were primarily evaluated through controlled experiments. Controlled experiments eliminate the influence of extraneous and unwanted variables that could account for a change of the dependent variable(s) other than the independent variable(s) of interest. Thus, their two main advantages are: (1) they are the most powerful method for inferring causal relationships between variables, and (2) they can achieve high internal validity [109]. Nevertheless, their main disadvantage is that since they are typically conducted in an artificial environment, conclusions may not be valid in real-life settings, which threatens the external validity [109].

Most studies that used controlled experiments as an evaluation approach evaluated results derived from a proposed artifact against a manually created ground-truth set. The quality of the ground-truth set, however, determines the performance of machine-learning algorithms. The majority of the studies, thus, recruited multiple annotators for the labeling task to obtain a “reliable” ground-truth set, which only contains peer-agreed labels. Some studies used an annotation guideline, performed a pilot run of classification tasks with small samples to avoid subjective assessment, reduce disagreements, and increase the quality of manual labeling [38, 39, 68].

Besides, a few studies compared the performance of automated analysis with a proposed artifact with the performance achieved by solely relying on manual analysis without the aid of the proposed artifact. Groen et al. [57] justified the efficiency and scalability of automated user review analysis and emphasized the need of automation for analyzing a large volume of dynamic data to support continuous requirements engineering. A case study was the second most frequently used evaluation approach in which the proposed methods are assessed through in-depth investigations of a specific instance in a real-life context. Proof of concept was used in a small proportion of the selected studies. It is used to demonstrate the feasibility of a proposed artifact theoretically to achieve an intended task. Although it may be suitable as a preliminary or formative evaluation, it has lower explanatory power compared to comparative evaluations (e.g., controlled experiments and case studies).

Most studies used standard metrics that are often used in the field of information retrieval. Completeness and correctness were the evaluation concepts that were the most frequently used in the studies, while some studies also assessed the efficiency of an artifact. Recall and precision were often used as metrics to measure completeness and correctness, respectively. Since there is a trade-off between precision and recall, many studies additionally used F-measure, which is the weighted harmonic mean of precision and recall. Most of the studies used F_1-measure, which assigns equal weights on precision and recall (i.e., the harmonic mean of precision and recall). However, Guzman et al. [69] recall was assigned more importance (i.e., weights) than precision based on the study which claims that recall should be favored over precision since missing relevant requirements is more detrimental [110]. On the other hand, precision is also important when dealing with a dataset that contains large amounts of irrelevant information. Future research may explore techniques to optimize F-measures, including a weighted maximum likelihood solution [111]. Moreover, few studies have compared the effectiveness of automated requirements elicitation with that of traditional requirements elicitation driven by stakeholders. This can largely be explained by the fact that research on automated requirements elicitation is not mature enough since most methods have focused on identifying and classifying requirements-related information rather than eliciting requirements. However, this needs to be addressed in future research to demonstrate the value of automated requirements elicitation.

RQ3: What Are the Outcomes of Automated Requirements Elicitation?

Expression of Requirements Elicitation

In traditional requirements engineering, requirements elicitation begins with the identification of relevant requirements sources such as stakeholders and domain documents, which is followed by two other sub-activities: the elicitation of existing requirements from the identified sources and elicitation of new and innovative requirements [1].

On the other hand, dynamic data-driven requirements elicitation has been done in the form of the following three activities: (1) identification and classification of requirements-related information, (2) identification of candidate features related to requirements, and (3) elicitation of requirements. However, those three activities have not necessarily been performed entirely nor sequentially. For example, many studies that aim to identify candidate features first performed classification, using supervised learning or rule-based classification, before clustering features, using topic modeling or traditional clustering, while the rest of them directly identified candidate software features, mainly using topic modeling or rule-based classification. One possible reason for performing classification before clustering is that classification can only classify data into coarse categories, which may include the repetitive information and the same sentiment, while clustering can further group individual data in a meaningful way. Thus, the specific combination of the two approaches can facilitate the work of requirements engineers (e.g., requirements reuse).

Most of the proposed methods supported the identification and classification of requirements-related information or the identification of candidate features. Identification and classification of requirements-related information help requirements engineers save time for the data analysis by filtering out a significant amount of irrelevant data and selectively identify a specific type of information which they are interested in such as feature request. It also helps to allocate the extracted data based on the relevance to stakeholder groups to support parallel data analysis within the same organization. Identification of candidate features helps requirements engineers understand user-preferred features and select features to be considered in software development and evolution. Features, however, are not yet formulated as requirements because those features require the engagement of requirements engineers to transform into requirements.

On the other hand, only about 20% of the studies automated the entire requirements elicitation. In most cases, the elicited requirements are high-level such as goals, aggregated scenarios, or high-level textual requirements. Those high-level requirements, however, do not include details of the objects (i.e., features) which are being concerned, nor conditions. This highlights the need for developing additional automated approaches or using traditional elicitation techniques with the involvement of human stakeholders to complete the requirements elicitation process.

Degree of Intended Automation

A majority of the studies proposed methods that are intended to be fully automated after the minimum human interventions at the initial stage of the continuous elicitation process. However, most studies do not yet support the entire requirements elicitation. Given the high volume and velocity of dynamic data, requirements elicitation certainly needs to be automated to enhance efficiency and scalability.

However, fully automated methods are not necessarily better than semi-automated methods concerning the quality of requirements and the ease of implementing into an existing requirements engineering process as well as the organizational workflow. There is a lack of evidence on what level of automation leads to the most effective requirements elicitation within an organization. More research, thus, needs to be done on whether it is possible and better to automate the entire elicitation process, or whether some extent of human-in-the-loop is necessary.

If a semi-automated approach is considered preferable, another issue that needs to be addressed is where and when in the elicitation process human should come into play to facilitate effective automated requirements elicitation. In addition, the characteristics of dynamic data can be changed over time. The proposed automated approach should be flexible enough to incorporate and reflect these dynamic changes over time.

Additional Requirements Engineering Activity Supported Through Automation

Our results show that three-quarters of the selected studies exclusively focused on requirements elicitation, while only one-quarter supported additional requirements engineering activities, which were requirements prioritization and management of requirements change. Therefore, no studies supported the automation of the entire requirements engineering process. A holistic framework, therefore, needs to be developed to increase the automation level of dynamic data-driven requirements engineering.

Threats to Validity

The results of the review need to be interpreted with caution due to the following limitations.

1. External validity

   All the studies included in the review, except one utilizing user feedback in both English and Italian [65], focus on eliciting English requirements. Thus, our results cannot be generalizable to requirements elicitation in other languages. Further studies are needed to assess the applicability of the techniques used for eliciting English requirements to other languages.
2. Internal validity

   Our search query might have missed potentially important keywords such as “requirements mining”, “feedback”, and “tool”. Not including those keywords affects the number of studies included in the analysis. Our search query also failed to capture the work following DevOps and human–computer interaction approaches, which may have resulted in omitting some important work. We did not perform a backward and forward reference search for updating the review. The absence also may have reduced the number of studies included in this review.

   In addition, a single reviewer performed a large part of study selection and data extraction, which may cause errors that impact the results. We partially mitigate the risk by ensuring high inter-rater reliability tested on a small proportion of randomly selected samples and discussing with at least one of the other reviewers to decide the inclusion of undecided papers, as explained in the “Study Selection” section. Ideally, the entire study selection and data extraction process should have been performed by at least two reviewers.

   Another limitation is that we defined an analytical framework to synthesize retrieved data in advance. However, the analytical framework was based on the previous systematic review of the automated requirements elicitation from domain documents. Moreover, we assessed the quality of individual study solely based on the SJR or CORE scores. Those scores may not always reflect the “true” strength of evidence provided by each study. A more detailed and formal quality assessment could have added value to the review by increasing the reliability of the results.
3. Publication bias

   This review included only published peer-reviewed studies and excluded gray literature and commercial products, which may fill many of the gaps identified in this review. Thus, the frequencies of the techniques and concepts do not imply real-life usage frequencies or degree of usefulness. Including gray literature and commercial products would increase the review’s completeness and timeliness.

Conclusions and Future Work

We have conducted a systematic literature review concerning requirements elicitation from data generated via digital technologies that are unintended with respect to requirements. These sources can include data that is highly relevant for new system requirements, which otherwise could not be obtained from other sources. The motivation behind the proposed approaches lies in the fact that by including such requirements, which existing or new software systems are not supporting, important improvements concerning system functionality and quality can be made, as well as ensuring that requirements are up-to-date and enabling further automation of a continuous elicitation process.

This literature review provides an overview of the state-of-the-art with respect to data-driven requirements elicitation from dynamic data sources. This is the first systematic review focusing on efforts to automate or support requirements elicitation from these types of data sources—often referred to as Big Data—that include not only human-sourced data but also process-mediated and machine-generated data.

We obtained 1848 relevant studies by searching six electronic databases. After two levels of screening, and a complementary forward and backward reference search, 51 papers were selected for data analysis. We further performed additional 2-level screening to update our search, which resulted in including 17 more studies. Thus, in total, 68 studies are included in the final analysis. Those selected studies were analyzed to answer the defined research questions concerning (a) identification of specific data sources and data types used for the elicitation, (b) methods and techniques used for processing the data, and (c) classification of the content of obtained outputs in relation to what is expected from the traditional elicitation process.

The results revealed remarkable insights, which, when summarized, have shown the current clear dominance of the human-sourced data, compared to the process-mediated and machine-generated data sources. As a result of that the techniques used for data processing are based on natural language processing, while the use of machine learning for classification and clustering is prevalent. The dominant intention of the proposed methods was to automate the elicitation process fully, rather than to combine it with traditional stakeholder-involved approaches.

Furthermore, the results showed that the majority of the studies were considering both functional and non-functional (i.e., quality) requirements. The final results regarding the completeness and the readiness of the elicited data for use in system development or evolution are currently limited—most of the studies obtain some of the information relevant for requirement's content, some studies target the identification of the core functionality or quality in terms of features, and only a few of the studies achieve a high-level requirement content. Finally, the majority of the studies evaluated the results in experimental environments, thus indicating rather a low extent of implementation of the method in a real-life requirements engineering setting.

The obtained results provide several directions for future work. One possible direction concerns the investigation of more extensive use and analysis of non-human-sourced data types. In addition, automatic data fusion and contextualization methods need to be investigated for integrating, processing, and analyzing a large volume of heterogeneous data sources to elicit requirements. Semantic technologies can be a promising solution to address the variety and volume issues of Big Data. Other direction leads to enabling real-time data processing and analyzing to facilitate continuous requirements elicitation from Big Data with high velocity.

Moreover, each proposed solution needs to be evaluated against traditional requirements to convince practitioners for its implementation in real-life. Further improvements also need to be made in the content and quality of the elicited data in relation to fully detailed requirements. Finally, a very important direction relates to the proposals for enabling context-awareness to capture requirements that changes dynamically over time.

Funding Open Access funding provided by Stockholm University. This study was funded by the Department of Computer and System Sciences, Stockholm University.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

Appendix 1

See Table 9

J1: J. A. Khan, L. Liu, and L. Wen. "Requirements knowledge acquisition from online user forums." *IET Software*, vol. 14, no. 3, pp. 242–253, https://doi.org/10.1049/iet-sen.2019.0262.

J2: M. Oriol et al., "Data-driven and tool-supported elicitation of quality requirements in agile companies," *Software Quality Journal*, pp. 1–33, 2020, https://doi.org/10.1007/s11219-020-09509-y.

J3: N. Ali, S. Hwang, and J.-E. Hong. "Your opinions let us know: Mining social network sites to evolve software product lines." *KSII Transactions on Internet and Information Systems*, vol. 13, no. 8, pp. 4191–4211, 2019, https://doi.org/10.3837/tiis.2019.08.021.

J4: A. Alwadain and M. Alshargi. "Crowd-generated data mining for continuous requirements elicitation." *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 9, pp. 45–50, 2019.

J5: Y. Kang, H. Li, C. Lu, and B. Pu. "A transfer learning algorithm for automatic requirement model generation." *Journal of Intelligent and Fuzzy Systems*, vol. 36, no. 2, pp. 1183–1191, 2019, https://doi.org/10.3233/JIFS-169892.

J6: H. Voet, M. Altenhof, M. Ellerich, R. H. Schmitt, and B. Linke. "A Framework for the capture and analysis of product usage data for continuous product improvement." *Journal of Manufacturing Science and Engineering,* vol.141, no. 2, 2019.

J7: L. Zhao and A. Zhao. "Sentiment analysis based requirement evolution prediction." *Future Internet*, vol. 11, no. 2, p. 52, 2019.

Table 9 General characteristics of the selected studies

No.	Authors	Year	Publication sources
J1	Khan et al. [80]	2020	IET Software
J2	Oriol et al. [89]	2020	Software Quality Journal
J3	Ali et al. [66]	2019	KSII Transactions on Internet and Information Systems
J4	Alwadain and Alshargi [67]	2019	International Journal of Advanced Computer Science and Applications (IJACSA)
J5	Kang et al. [91]	2019	Journal of Intelligent & Fuzzy Systems
J6	Voet et al. [95]	2019	Journal of Manufacturing Science and Engineering
J7	Zhao and Zhao [47]	2019	Future Internet
J8	Jha and Mahmoud [48]	2018	Empirical Software Engineering
J9	Morales-Ramirez et al. [84]	2018	Information Systems
J10	Guzman et al. [69]	2017	Requirements Engineering
J11	Xie et al. [97]	2017	Journal of Systems and Software
J12	Bakar et al. [61]	2016	Applied Soft Computing
J13	Licorish [102]	2016	Journal of Software
J14	Maalej et al. [38]	2016	Requirements Engineering
J15	Vlas and Robinson [76]	2012	Journal of Management Information Systems
J16	Cleland-Huang et al. [78]	2009	Communications of the ACM
C1	Dalpiaz and Parente [46]	2019	International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)
C2	Do et al. [90]	2019	International Conference on Software and Systems Reuse (ICSR)
C3	Han et al. [75]	2019	IEEE International Conference on Automation and Computing (ICAC)
C4	Khan [82]	2019	International Requirements Engineering Conference (RE)
C5	Khan et al. [81]	2019	International Requirements Engineering Conference (RE)
C6	Kilani et al. [104]	2019	International Conference on Social Networks Analysis, Management and Security (SNAMS)
C7	Martens and Maalej [74]	2019	International Requirements Engineering Conference (RE)
C8	Nyamawe et al. [87]	2019	International Requirements Engineering Conference (RE)
C9	Tizard et al. [83]	2019	International Requirements Engineering Conference (RE)
C10	Wang et al. [93]	2019	International Conference on Software Engineering and Knowledge Engineering (SEKE)
C11	Wüest et al. [99]	2019	International Working Conference on Requirements Engineering (REFSQ)
C12	Buchan et al. [59]	2018	Australasian Software Engineering Conference (ASWEC)
C13	Dhinakaran et al. [35]	2018	IEEE International Requirements Engineering Conference (RE)
C14	Franch et al. [88]	2018	International Conference on Advanced Information Systems Engineering (CAiSE)
C15	Groen et al. [57]	2018	International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)
C16	Higashi et al. [54]	2018	International Conference on Software Engineering & Knowledge Engineering (SEKE)
C17	Luiz et al. [49]	2018	International World Wide Web Conference (WWW)
C18	Srisopha et al. [56]	2018	CIbSE XXI Ibero-American Conference on Software Engineering (CIbSE)
C19	Bakar et al. [63]	2017	International Conference on Computing and Informatics (ICOCI)
C20	Lu and Liang [42]	2017	International Conference on Evaluation and Assessment in Software Engineering (EASE)
C21	Groen et al. [37]	2017	IEEE International Requirements Engineering Conference (RE)
C22	Guzman et al. [68]	2017	IEEE International Requirements Engineering Conference (RE)
C23	Jha and Mahmoud [51]	2017	International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)
C24	Morales-Ramirez et al. [79]	2017	International Conference on Advanced Information Systems Engineering (CAiSE)
C25	Williams and Mahmoud [64]	2017	IEEE International Requirements Engineering Conference (RE)
C26	Johann et al. [94]	2017	IEEE International Requirements Engineering Conference (RE)
C27	Yang et al. [98]	2017	Annual Computer Software and Applications Conference (COMPSAC)
C28	Guzman et al. [70]	2016	IEEE International Requirements Engineering Conference (RE)
C29	Kuehl [71]	2016	International Conference on Exploring Services Science (IESS)
C30	Merten et al. [85]	2016	IEEE International Requirements Engineering Conference (RE)
C31	Nguyen et al. [72]	2016	IFIP Working Conference on The Practice of Enterprise Modeling (PoEM)

Table 9 (continued)

No.	Authors	Year	Publication sources
C32	Svee and Zdravkovic [73]	2016	International Conference on Research Challenges in Information Science (RCIS)
C33	Bakar et al. [62]	2015	International Conference on Information Science and Security (ICISS)
C34	Maalej and Nabil [39]	2015	IEEE International Requirements Engineering Conference (RE)
C35	Panichella et al. [45]	2015	IEEE International Conference on Software Maintenance and Evolution (ICSME)
C36	Takahashi et al. [100]	2015	International Conference on Software Engineering & Knowledge Engineering (SEKE)
C37	Sun and Peng [53]	2015	Asia Pacific Requirements Engineering Symposium (APRES)
C38	Guzman and Maalej [40]	2014	IEEE International Requirements Engineering Conference (RE)
C39	Jiang et al. [41]	2014	Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)
C40	Zhang et al. [60]	2014	International Conference on Enterprise Systems (ES)
C41	Carreño and Winbladh [52]	2013	International Conference on Software Engineering (ICSE)
W1	Stanik et al. [65]	2019	International Requirements Engineering Conference Workshops (REW)
W2	Do and Bhowmik [36]	2018	ACM SIGSOFT International Workshop on Automated Specification Inference
W3	Abad et al. [44]	2017	International Requirements Engineering Conference Workshops (REW)
W4	Bakiu and Guzman [55]	2017	International Requirements Engineering Conference Workshops (REW)
W5	Deocadez et al. [43]	2017	International Requirements Engineering Conference Workshops (REW)
W6	Jha and Mahmoud [50]	2017	REFSQ Workshops
W7	Portugal et al. [86]	2015	IEEE Workshop on Just-In-Time Requirements Engineering (JITRE)
S1	Wang et al. [92]	2018	ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)
S2	Liang et al. [96]	2015	Asia–Pacific Symposium on Internetware
S3	Xiao et al. [77]	2015	Communications in Computer and Information Science (CCIS)
S4	Jiang et al. [58]	2014	Communications in Computer and Information Science (CCIS)

J8: N. Jha and A. Mahmoud. "Using frame semantics for classifying and summarizing application store reviews." *Empirical Software Engineering* 23, no. 6 (2018): 3734–3767.

J9: I. Morales-Ramirez, F. M. Kifetew, and A. Perini, "Speech-acts based analysis for requirements discovery from online discussions," *Information Systems*, vol. 86, pp.94–112, 2018, https://doi.org/10.1016/j.is.2018.08.003.

J10: E. Guzman, R. Alkadhi, and N. Seyff. "An exploratory study of Twitter messages about software applications." *Requirements Engineering*, vol. 22, no. 3, pp. 387–412, 2017.

J11: H. Xie, J. Yang, C. K. Chang, and L. Liu. "A statistical analysis approach to predict user's changing requirements for software service evolution." *Journal of Systems and Software*, vol. 132, pp. 147–164, 2017, https://doi.org/10.1016/j.jss.2017.06.071.

J12: N. H. Bakar, Z. M. Kasirun, N. Salleh, and H. A. Jalab. "Extracting features from online software reviews to aid requirements reuse." *Applied Soft Computing Journal*, vol. 49, pp. 1297–1315, 2016, https://doi.org/10.1016/j.asoc.2016.07.048.

J13: S. A. Licorish. "Exploring the prevalence and evolution of Android concerns: A community viewpoint." *JSW*, vol. 11, no. 9, pp. 848–869, 2016.

J14: 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, https://doi.org/10.1007/s00766-016-0251-9.

J15: R. E. Vlas and W. N. Robinson. "Two rule-based natural language strategies for requirements discovery and classification in open source software Development Projects." *Journal of Management Information Systems*, vol. 28, no. 4, pp. 11–38,, https://doi.org/10.2753/MIS0742-1222280402.

J16: J. Cleland-Huang, H. Dumitru, C. Duan, and C. Castro-Herrera. "Automated support for managing feature requests in open forums." *Communications of the ACM*, vol. 52, no. 10, pp. 68–74, 2009, https://doi.org/10.1145/1562764.1562784.

C1: F. Dalpiaz and M. Parente. "RE-SWOT: From user feedback to requirements via competitor analysis." In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 55–70. Springer, Cham, 2019.

C2: Q. A. Do, S. R. Chekuri, and T. Bhowmik. "Automated support to capture creative requirements via requirements reuse." In *International Conference on Software and Systems Reuse*, pp. 47–63. Springer, Cham, 2019, https://doi.org/10.1007/978-3-030-22888-0_4.

C3: X. Han, R. Li, W. Li, G. Ding, and S. Qin. "User requirements dynamic elicitation of complex products from

social network service." In *2019 25th International Conference on Automation and Computing (ICAC)*, pp. 1–6. IEEE, 2019. https://doi.org/10.23919/IConAC.2019.8895140.

C4: J. A. Khan. "Mining requirements arguments from user forums." In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 440–445. IEEE, 2019.

C5: J. A. Khan, Y. Xie, L. Liu, and L. Wen. "Analysis of requirements-related arguments in user forums." In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 63–74. IEEE, 2019.

C6: N. Al Kilani, R. Tailakh, and A. Hanani. "Automatic classification of apps reviews for requirement engineering: exploring the customers need from healthcare applications." In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pp. 541–548. IEEE, 2019, https://doi.org/10.1109/SNAMS.2019.8931820.

C7: D. Martens and W. Maalej. "Extracting and analyzing context information in user-support conversations on twitter," In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 131–141. IEEE, 2019.

C8: A. S. Nyamawe, H. Liu, N. Niu, Q. Umer, and Z. Niu. "Automated recommendation of software refactorings based on feature requests." In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 187–198. IEEE, 2019.

C9: J. Tizard, H. Wang, L. Yohannes, and K. Blincoe. "Can a conversation paint a picture? Mining requirements in software forums." In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 17–27. IEEE, 2019.

C10: C. Wang, T. Wang, P. Liang, M. Daneva, and M. Van Sinderen. "Augmenting app review with app changelogs: An approach for app review classification." In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 398–512. 2019, https://doi.org/10.18293/SEKE2019-176.

C11: D. Wüest, F. Fotrousi, and S. Fricker. "Combining monitoring and autonomous feedback requests to elicit actionable knowledge of system use." In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 209–225. Springer, Cham, 2019.

C12: J. Buchan, M. Bano, D. Zowghi, and P. Volabouth. "Semi-automated extraction of new requirements from online reviews for software product evolution." In *2018 25th Australasian Software Engineering Conference (ASWEC)*, pp. 31–40. IEEE, 2018.

C13: V. T. Dhinakaran, R. Pulle, N. Ajmeri, and P. K. Murukannaiah. "App review analysis via active learning: reducing supervision effort without compromising classification accuracy." In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pp. 170–181. IEEE, 2018, https://doi.org/10.1109/RE.2018.00026.

C14: X. Franch et al.. "Data-driven elicitation, assessment and documentation of quality requirements in agile software development." In *International Conference on Advanced Information Systems Engineering*, pp. 587–602. Springer, Cham, 2018.

C15: E. C. Groen, F. Iese, J. Schowalter, and S. Kopczynska. "Is there really a need for using NLP to elicit requirements? A benchmarking study to assess scalability of manual Analysis." In *REFSQ Workshops*. 2018.

C16: K. Higashi, H. Nakagawa, and T. Tsuchiya. "Improvement of user review classification using keyword expansion." In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 125–124. 2018.

C17: W. Luiz et al.. "A feature-oriented sentiment rating for mobile app reviews." In *Proceedings of the 2018 World Wide Web Conference*, pp. 1909–1918. 2018.

C18: K. Srisopha, P. Behnamghader, and B. Boehm. "Do users talk about the software in my product? Analyzing user reviews on IoT products." In *the Proceedings of CIbSE XXI Ibero-American Conference on Software Engineering (CIbSE)*, pp.551–564, 2018

C19: N. H. Bakar, Z. M. Kasirun, N. Salleh, and A. Halim. "Extracting software features from online reviews to demonstrate requirements reuse in software engineering." In *Proceedings of the International Conference on Computing & Informatics*, pp. 184–190. 2017.

C20: M. Lu and P. Liang. "Automatic classification of non-functional requirements from augmented app user reviews." In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 2017, pp. 344–353.

C21: E. C. Groen, S. Kopczynska, 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 *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 80–89. IEEE, 2017, https://doi.org/10.1109/RE.2017.73.

C22: E. Guzman, M. Ibrahim, and M. Glinz. "A little bird told me: Mining tweets for requirements and software evolution." In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 11–20. IEEE, 2017, https://doi.org/10.1109/RE.2017.88.

C23: N. Jha and A. Mahmoud. "Mining user requirements from application store reviews using frame semantics." In *International working conference on requirements engineering: Foundation for software quality*, pp. 273–287. Springer, Cham, 2017.

C24: I. Morales-Ramirez, F. M. Kifetew, and A. Perini. "Analysis of online discussions in support of requirements discovery." In *International Conference on Advanced Information Systems Engineering (CAiSE)*, pp. 159–174. Springer, Cham, 2017, https://doi.org/10.1007/978-3-319-59536-8_11.

C25: G. Williams and A. Mahmoud. “Mining twitter feeds for software user requirements.” In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 1–10. IEEE, 2017, https://doi.org/10.1109/RE.2017.14.

C26: T. Johann, C. Stanik, A. M. A. B, and W. Maalej. “SAFE: A simple approach for feature extraction from app descriptions and app reviews.” In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 21–30. IEEE, 2017, https://doi.org/10.1109/RE.2017.71.

C27: J. Yang, C. K. Chang, and H. Ming. “A situation-centric approach to identifying new user intentions using the mtl method.” In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 347–356. IEEE, 2017.

C28: E. Guzman, R. Alkadhi, and N. Seyff. “A needle in a haystack: What do twitter users say about software?,” In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pp. 96–105. IEEE, 2016, https://doi.org/10.1109/RE.2016.67.

C29: N. Kuehl. "Needmining: Towards analytical support for service design." In *International Conference on Exploring Services Science*, pp. 187–200. Springer, Cham, 2016.

C30: T. Merten, M. Falis, P. Hübner, T. Quirchmayr, S. Bürsner, and B. Paech. “Software feature request detection in issue tracking systems.” In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pp. 166–175. IEEE, 2016,, https://doi.org/10.1109/RE.2016.8.

C31: V. Nguyen, E. Svee, and J. Zdravkovic. "A semi-automated method for capturing consumer preferences for system requirements." In *IFIP Working Conference on The Practice of Enterprise Modeling*, pp. 117–132. Springer, Cham, 2016.

C32: E. Svee and J. Zdravkovic. "A model-based approach for capturing consumer preferences from crowdsources: the case of Twitter." In *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, pp. 1–12. IEEE, 2016, https://doi.org/10.1109/RCIS.2016.7549323.

C33: N. H. Bakar, Z. M. Kasirun, and N. Salleh. “Terms extractions: An approach for requirements reuse.” In *2015 2nd International Conference on Information Science and Security (ICISS)*, pp. 1–4. IEEE, 2015.–254. IEEE, 2015, https://doi.org/10.1109/ICISSEC.2015.7371034.

C34: W. Maalej and H. Nabil. "Bug report, feature request, or simply praise? on automatically classifying app reviews." In *2015 IEEE 23rd international requirements engineering conference (RE)*, pp. 116–125. IEEE, 2015, https://doi.org/10.1109/RE.2015.7320414.

C35: S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall. "How can I improve my app? Classifying user reviews for software maintenance and evolution." In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 281–290. IEEE, 2015, https://doi.org/10.1109/ICSM.2015.7332474.

C36: H. Takahashi, H. Nakagawa, and T. Tsuchiya, “Towards automatic requirements elicitation from feedback comments: Extracting requirements topics using LDA,” In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 489–494. 2015, https://doi.org/10.18293/SEKE2015-103.

C37: D. Sun and R. Peng. “A scenario model aggregation approach for mobile app requirements evolution based on user comments.” In *Requirements Engineering in the Big Data Era*, pp. 75–91. Springer, Berlin, Heidelberg, 2015.

C38: E. Guzman and W. Maalej. “How do users like this feature? A fine grained sentiment analysis of app reviews.” In *2014 IEEE 22nd international requirements engineering conference (RE)*, pp. 153–162. IEEE, 2014, https://doi.org/10.1109/RE.2014.6912257.

C39: W. Jiang, H. Ruan, L. Zhang, P. Lew, and J. Jiang. “For user-driven software evolution: requirements elicitation derived from mining online reviews.” In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 584–595. Springer, Cham, 2014.

C40: Z. Zhang, J. Qi, and G. Zhu. “Mining customer requirement from helpful online reviews,” In *2014 Enterprise Systems Conference*, pp. 249–254. IEEE, 2014, https://doi.org/10.1109/ES.2014.38.

C41: L. V. G. Carreno and K. Winbladh. “Analysis of user comments: An approach for software requirements evolution” In *2013 35th international conference on software engineering (ICSE)*, pp. 582–591. IEEE, 2013, https://doi.org/10.1109/ICSE.2013.6606604.

W1: C. Stanik, M. Haering, and W. Maalej. “Classifying multilingual user feedback using traditional machine learning and deep learning.” In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pp. 220–226. IEEE, 2019.

W2: Q. A. Do and T. Bhowmik. "Automated generation of creative software requirements: a data-driven approach." In *Proceedings of the 1st ACM SIGSOFT International Workshop on Automated Specification Inference*, pp. 9–12. 2018, https://doi.org/10.1145/3278177.3278180.

W3: Z. S. H. Abad, S. D. V. Sims, A. Cheema, M. B. Nasir, and P. Harisinghani. “Learn more, pay less! Lessons learned from applying the wizard-of-oz technique for exploring mobile app requirements.” In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, Sep. 2017, pp. 132–138, https://doi.org/10.1109/REW.2017.71.

W4: E. Bakiu and E. Guzman. “Which feature is unusable? Detecting usability and user experience issues from user reviews.” In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pp. 182–187. IEEE, 2017, https://doi.org/10.1109/REW.2017.76.

Table 10 Quality assessment of the included studies

No	SJR	CORE	Point
J1	Q3	–	1
J2	Q2	–	1.5
J3	Q3	–	1
J4	Q4	–	1
J5	Q2	–	1.5
J6	Q1	–	2
J7	Q2	–	1.5
J8	Q1	–	1
J9	Q1	–	1
J10	Q1	–	1
J11	Q1	–	1
J12	Q1	–	1
J13	No	–	0
J14	Q1	–	2
J15	Q1	–	2
J16	Q1	–	2
C1	–	B	1
C2	–	Unlisted	0
C3	–	Unlisted	0
C4	–	A	1.5
C5	–	A	1.5
C6	–	Unlisted	0
C7	–	A	1.5
C8	–	A	1.5
C9	–	A	1.5
C10	–	B	1
C11	–	B	1
C12	–	Unlisted	0
C13	–	A	1.5
C14	–	A	1.5
C15	–	B	1
C16	–	B	1
C17	–	A*	1.5
C18	–	Unlisted	0
C19	–	C	0.5
C20	–	A	1.5
C21	–	A	1.5
C22	–	A	1.5
C23	–	B	1
C24	–	A	1.5
C25	–	A	1.5
C26	–	A	1.5
C27	–	Unlisted	0
C28	–	A	1.5
C29	–	Unlisted	0
C30	–	A	1.5
C31	–	Unlisted	0
C32	–	B	1
C33	–	Unlisted	0

Table 10 (continued)

No	SJR	CORE	Point
C34	–	A	1.5
C35	–	A	1.5
C36	–	B	1
C37	–	Unlisted	0
C38	–	A	1.5
C39	–	A	1.5
C40	–	Unlisted	0
C41	–	A*	1.5
W1	–	Unlisted	0
W2	–	Unlisted	0
W3	–	Unlisted	0
W4	–	Unlisted	0
W5	–	Unlisted	0
W6	–	Unlisted	0
W7	–	Unlisted	0
S1	–	A	1.5
S2	–	Unlisted	0
S3	–	Unlisted	0
S4	–	Unlisted	0
Median			1
Q3			1.5
Q1			0

W5: R. Deocadez, R. Harrison, and D. Rodriguez. "Automatically classifying requirements from app stores: A preliminary study." In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, Sep. 2017, pp. 367–371, https://doi.org/10.1109/REW.2017.58.

W6: N. Jha and A. Mahmoud. "MARC: A mobile application review classifier." In *REFSQ Workshops*. 2017.

W7: R. L. Q. Portugal, J. C. S. Do Prado Leite, and E. Almentero. "Time-constrained requirements elicitation: Reusing GitHub content." In *2015 IEEE Workshop on Just-In-Time Requirements Engineering (JITRE)*, pp. 5–8. IEEE, 2015, https://doi.org/10.1109/JITRE.2015.7330171.

S1: C. Wang, F. Zhang, P. Liang, M. Daneva, and M. van Sinderen. "Can app changelogs improve requirements classification from app reviews?: An exploratory study." In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 1–4. 2018, https://doi.org/10.1145/3239235.3267428.

S2: W. Liang, W. Qian, Y. Wu, X. Peng, and W. Zhao. "Mining context-aware user requirements from crowd contributed mobile data." In *Proceedings of the 7th Asia–Pacific Symposium on Internetware*, pp. 132–140. 2015, https://doi.org/10.1145/2875913.2875933.

S3: M. Xiao, G. Yin, T. Wang, C. Yang, and M. Chen. "Requirement acquisition from social Q&A sites." In *Liu L., Aoyama M. (eds) Requirements Engineering in the Big Data Era. Communications in Computer and Information Science*, vol 558. Springer, Berlin, Heidelberg, 2015.

S4: W. Jiang, H. Ruan, and L. Zhang. "Analysis of economic impact of online reviews: An approach for market-driven requirements evolution." In Zowghi D., Jin Z. (eds) Requirements Engineering. Communications in Computer and Information Science, vol. 432, pp. 45–59, 2014, Springer, Berlin, Heidelberg, https://doi.org/10.1007/978-3-662-43610-3_4.

Appendix 2

See Table 10

References

1. Pohl K. Requirements engineering: fundamentals, principles, and techniques. Heidelberg: Springer; 2010.
2. Pacheco C, García I, Reyes M. Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques. IET Softw. 2018;12(4):365–78. https://doi.org/10.1049/iet-sen.2017.0144.
3. Chen H, Chiang RHL, Storey VC. Business intelligence and analytics: from big data to big impact. MIS Quart. 2012;36(4):1165–88.

4. Manrique-Losada B, Zapata-Jaramillo CM, Burgos DA. Re-expressing business processes information from corporate documents into controlled language. In: *International Conference on Applications of Natural Language Processing to Information Systems*. Cham: Springer, 2016, pp. 376–383.
5. Hauksdóttir D, Ritsing B, Andersen JC, Mortensen NH. Establishing reusable requirements derived from laws and regulations for medical device development. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, 2016, pp. 220–228, https://doi.org/10.1109/REW.2016.045.
6. Kaiya H, Saeki M. Using domain ontology as domain knowledge for requirements elicitation. In *14th IEEE International Requirements Engineering Conference (RE'06)*, pp. 189–198. IEEE, 2006, pp. 186–195, https://doi.org/10.1109/RE.2006.72.
7. Zong-yong L, Zhi-xue W, Ying-ying Y, Yue W, Ying L. Towards a multiple ontology framework for requirements elicitation and reuse. In *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, vol. 1, pp. 189–195. IEEE, 2007, https://doi.org/10.1109/COMPSAC.2007.216.
8. Nogueira FA, De Oliveira HC. Application of heuristics in business process models to support software requirements specification. ICEIS. 2017;2:40–51.
9. Bendjenna H, Zarour NE, Charrel P. MAMIE: A methodology to elicit requirements in inter-company co-operative information systems. In *2008 International Conference on Computational Intelligence for Modelling Control & Automation*, 2008, pp. 290–295, https://doi.org/10.1109/CIMCA.2008.101.
10. Shao F, Peng R, Lai H, Wang B. DRank: a semi-automated requirements prioritization method based on preferences and dependencies. J Syst Softw. 2017;126:141–56. https://doi.org/10.1016/j.jss.2016.09.043.
11. Abad ZSH, Karras O, Ghazi P, Glinz M, Ruhe G, Schneider K. "What works better? A study of classifying requirements. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 496–501. IEEE, 2017, https://doi.org/10.1109/RE.2017.36.
12. Hayes JH, Antoniol G, Adams B, Guehéneuc YG. Inherent characteristics of traceability artifacts: Less is more. In: *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pp. 196–201. IEEE, 2015.
13. Kamalrudin M, Hosking J, Grundy J. MaramaAIC: tool support for consistency management and validation of requirements. Automat Softw Eng. 2017;24(1):1–45. https://doi.org/10.1007/s10515-016-0192-z.
14. Ahmed MA, Butt WH, Ahsan I, Anwar MW, Latif M, Azam F. A novel natural language processing (NLP) approach to automatically generate conceptual class model from initial software requirements. In *International Conference on Information Science and Applications*, pp. 476–484. Springer, Singapore, 2017.
15. Kifetew F, Munante D, Perini A, Susi A, Siena A, Busetta P. DMGame: A gamified collaborative requirements prioritisation tool. In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 468–469, https://doi.org/10.1109/RE.2017.46.
16. Ahmad S, Jalil IEA, Ahmad SSS. An enhancement of software requirements negotiation with rule-based reasoning: a conceptual model. J Telecommun Electron Comput Eng (JTEC). 2016;8(10):193–8.
17. Meth H, Brhel M, Maedche A. The state of the art in automated requirements elicitation. Inf Softw Technol. 2013;55(10):1695–709. https://doi.org/10.1016/j.infsof.2013.03.008.
18. Nicolás J, Toval A. On the generation of requirements specifications from software engineering models: a systematic literature review. Inf Softw Technol. 2009;51(9):1291–307. https://doi.org/10.1016/j.infsof.2009.04.001.
19. Groen EC, et al. The crowd in requirements engineering: the landscape and challenges. IEEE Softw. 2017;34(2):44–52. https://doi.org/10.1109/MS.2017.33.
20. Ferguson M. Big data-why transaction data is mission critical to success. Intelligence Business Strategies Limited. https://public.dhe.ibm.com/common/ssi/ecm/im/en/iml14442usen/IML14442USEN.PDF, 2014.
21. Maalej W, Nayebi M, Johann T, Ruhe G. Toward data-driven requirements engineering. IEEE Softw. 2016;33(1):48–54. https://doi.org/10.1109/MS.2015.153.
22. Genc-Nayebi N, Abran A. A systematic literature review: opinion mining studies from mobile app store user reviews. J Syst Softw. 2017;125:207–19. https://doi.org/10.1016/j.jss.2016.11.027.
23. Ghasemi M, Amyot D. From event logs to goals: a systematic literature review of goal-oriented process mining. Requir Eng. 2019;25(1):67–93. https://doi.org/10.1007/s00766-018-00308-3.
24. Zowghi D, Coulin C. Requirements elicitation: A survey of technique, approaches and tools. In: Engineering and managing software requirements. Berlin: Springer; 2005. p. 19–46.
25. Arruda D, Madhavji NH. State of requirements engineering research in the context of Big Data applications. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Cham: Springer, 2018, pp. 307–323.
26. Martin W, Sarro F, Jia Y, Zhang Y, Harman M. A survey of app store analysis for software engineering. IEEE Trans Software Eng. 2017;43(9):817–47. https://doi.org/10.1109/TSE.2016.2630689.
27. Tavakoli M, Zhao L, Heydari A, Nenadić G. Extracting useful software development information from mobile application reviews: a survey of intelligent mining techniques and tools. Exp Syst Appl. 2018;113:186–99.
28. De Mauro A, Greco M, Grimaldi M. What is big data? A consensual definition and a review of key research topics. In: *AIP conference proceedings*, vol. 1644, no. 1, pp. 97–104. American Institute of Physics, 2015, https://doi.org/10.1063/1.4907823.
29. Morales-Ramirez I, Perini A, Guizzardi RSS. An ontology of online user feedback in software engineering. Appl Ontol. 2015;10(3–4):297–330. https://doi.org/10.3233/AO-150150.
30. Firmani D, Mecella M, Scannapieco M, Batini C. On the meaningfulness of 'big data quality' (invited paper). Data Sci Eng. 2016;1(1):6–20. https://doi.org/10.1007/s41019-015-0004-7.
31. Kitchenham B, Charters S. Guidelines for performing systematic literature reviews in software engineering. 2007.
32. Landis JR, Koch GG. The measurement of observer agreement for categorical data. Biometrics. 1977;33(1):159–74. https://doi.org/10.2307/2529310.
33. Rotondi MA, Donner A. A confidence interval approach to sample size estimation for interobserver agreement studies with multiple raters and outcomes. J Clin Epidemiol. 2012;65(7):778–84. https://doi.org/10.1016/j.jclinepi.2011.10.019.
34. Hevner AR, March ST, Park J, Ram S. Design science in information systems research. MIS Quart. 2004;28(1):75–105. https://doi.org/10.2307/25148625.
35. Dhinakaran VT, Pulle R, Ajmeri N, Murukannaiah PK. App review analysis via active learning: Reducing supervision effort without compromising classification accuracy. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pp. 170–181. IEEE, 2018, https://doi.org/10.1109/RE.2018.00026.
36. Do QA, Bhowmik T. Automated generation of creative software requirements: a data-driven approach. In *Proceedings of the 1st ACM SIGSOFT International Workshop on Automated Specification Inference*, pp. 9–12. 2018, https://doi.org/10.1145/3278177.3278180.
37. Groen EC, Kopczynska S, Hauer MP, Krafft TD, Doerr J. Users-the hidden software product quality experts?: A study on how

app users report quality aspects in online reviews. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 80–89. IEEE, 2017, https://doi.org/10.1109/RE.2017.73.
38. Maalej W, Kurtanović Z, Nabil H, Stanik C. On the automatic classification of app reviews. Requir Eng. 2016;21(3):311–31. https://doi.org/10.1007/s00766-016-0251-9.
39. Maalej W, Nabil H. Bug report, feature request, or simply praise? On automatically classifying app reviews. In *2015 IEEE 23rd international requirements engineering conference (RE)*, pp. 116–125. IEEE, 2015, https://doi.org/10.1109/RE.2015.7320414.
40. Guzman E, Maalej W. How do users like this feature? A fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)*, pp. 153–162. IEEE, 2014, https://doi.org/10.1109/RE.2014.6912257.
41. Jiang W, Ruan H, Zhang L, Lew P, Jiang J. For user-driven software evolution: Requirements elicitation derived from mining online reviews. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 584–595. Springer, Cham, 2014.
42. Lu M, Liang P. Automatic classification of non-functional requirements from augmented app user reviews. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 2017, pp. 344–353.
43. Deocadez R, Harrison R, Rodriguez D. Automatically classifying requirements from app stores: A preliminary study. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, Sep. 2017, pp. 367–371, https://doi.org/10.1109/REW.2017.58.
44. Abad ZSH, Sims SDV, Cheema A, Nasir MB, Harisinghani P. Learn more, pay less! Lessons learned from applying the wizard-of-oz technique for exploring mobile app requirements. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, Sep. 2017, pp. 132–138, https://doi.org/10.1109/REW.2017.71.
45. Panichella S, Di Sorbo A, Guzman E, Visaggio CA, Canfora G, Gall HC. How can I improve my app? Classifying user reviews for software maintenance and evolution. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 281–290. IEEE, 2015, https://doi.org/10.1109/ICSM.2015.7332474.
46. Dalpiaz F, Parente M. RE-SWOT: From user feedback to requirements via competitor analysis. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 55–70. Springer, Cham, 2019.
47. Zhao L, Zhao A. Sentiment analysis based requirement evolution prediction. Fut Internet. 2019;11(2):52.
48. Jha N, Mahmoud A. Using frame semantics for classifying and summarizing application store reviews. Empir Softw Eng. 2018;23(6):3734–67.
49. Luiz W, et al. A feature-oriented sentiment rating for mobile app reviews. In *Proceedings of the 2018 World Wide Web Conference*, pp. 1909–1918. 2018.
50. Jha N, Mahmoud A. MARC: a mobile application review classifier. In *REFSQ Workshops*. 2017.
51. Jha N, Mahmoud A. Mining user requirements from application store reviews using frame semantics. In *International working conference on requirements engineering: Foundation for software quality*, pp. 273–287. Springer, Cham, 2017.
52. Carreno LVG, Winbladh K. Analysis of user comments: an approach for software requirements evolution. In *2013 35th international conference on software engineering (ICSE)*, pp. 582–591. IEEE, 2013, https://doi.org/10.1109/ICSE.2013.6606604.
53. Sun D, Peng R. A scenario model aggregation approach for mobile app requirements evolution based on user comments. In: Requirements engineering in the big data era. Berlin: Springer; 2015. p. 75–91.
54. Higashi K, Nakagawa H, Tsuchiya T. Improvement of user review classification using keyword expansion. In: *Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 125–124. 2018.
55. Bakiu E, Guzman E. Which feature is unusable? Detecting usability and user experience issues from user reviews. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pp. 182–187. IEEE, 2017, https://doi.org/10.1109/REW.2017.76.
56. Srisopha K, Behnamghader P, Boehm B. Do users talk about the software in my product? Analyzing user reviews on IoT products. In *the Proceedings of CIbSE XXI Ibero-American Conference on Software Engineering (CIbSE)*, pp. 551–564, 2018.
57. Groen EC, Iese F, Schowalter J, Kopczynska S. Is there really a need for using NLP to elicit requirements? A benchmarking study to assess scalability of manual analysis. In REFSQ Workshops. 2018.
58. Jiang W, Ruan H, Zhang L. Analysis of economic impact of online reviews: an approach for market-driven requirements evolution. In: Zowghi D, Jin Z, editors. Requirements engineering communications in computer and information science, vol. 432. Berlin: Springer; 2014. p. 45–59. https://doi.org/10.1007/978-3-662-43610-3_4.
59. Buchan J, Bano M, Zowghi D, Volabouth P. Semi-automated extraction of new requirements from online reviews for software product evolution. In *2018 25th Australasian Software Engineering Conference (ASWEC)*, pp. 31–40. IEEE, 2018.
60. Zhang Z, Qi J, Zhu G. Mining customer requirement from helpful online reviews. *2014 Enterprise Systems Conference*, pp. 249–254. IEEE, 2014, https://doi.org/10.1109/ES.2014.38.
61. Bakar NH, Kasirun ZM, Salleh N, Jalab HA. Extracting features from online software reviews to aid requirements reuse. Appl Soft Comput J. 2016;49:1297–315. https://doi.org/10.1016/j.asoc.2016.07.048.
62. Bakar NH, Kasirun ZM, Salleh N. Terms extractions: an approach for requirements reuse. In *2015 2nd International Conference on Information Science and Security (ICISS)*, pp. 1–4. IEEE, 2015.–254. IEEE, 2015, https://doi.org/10.1109/ICISSEC.2015.7371034.
63. Bakar NH, Kasirun ZM, Salleh N, Halim A. Extracting software features from online reviews to demonstrate requirements reuse in software engineering. In *Proceedings of the International Conference on Computing & Informatics*, pp. 184–190. 2017.
64. Williams G, Mahmoud A. Mining twitter feeds for software user requirements. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 1–10. IEEE, 2017, https://doi.org/10.1109/RE.2017.14.
65. Stanik C, Haering M, Maalej W. Classifying multilingual user feedback using traditional machine learning and deep learning. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pp. 220–226. IEEE, 2019.
66. Ali N, Hwang S, Hong J-E. Your opinions let us know: mining social network sites to evolve software product lines. KSII Trans Internet Inf Syst. 2019;13(8):4191–211. https://doi.org/10.3837/tiis.2019.08.021.
67. Alwadain A, Alshargi M. Crowd-generated data mining for continuous requirements elicitation. Int J Adv Comput Sci Appl. 2019;10(9):45–50.
68. Guzman E, Ibrahim M, Glinz M. A little bird told me: Mining tweets for requirements and software evolution. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 11–20. IEEE, 2017, https://doi.org/10.1109/RE.2017.88.

69. Guzman E, Alkadhi R, Seyff N. An exploratory study of twitter messages about software applications. Requir Eng. 2017;22(3):387–412.
70. Guzman E, Alkadhi R, Seyff N. A needle in a haystack: What do twitter users say about software? In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pp. 96–105. IEEE, 2016, https://doi.org/10.1109/RE.2016.67.
71. Kuehl N. Needmining: towards analytical support for service design. In *International Conference on Exploring Services Science*, pp. 187–200. Springer, Cham, 2016.
72. Nguyen V, Svee EO, Zdravkovic J. A semi-automated method for capturing consumer preferences for system requirements. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pp. 117–132. Springer, Cham, 2016.
73. Svee EO, Zdravkovic J. A model-based approach for capturing consumer preferences from crowdsources: the case of twitter. In *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, pp. 1–12. IEEE, 2016, https://doi.org/10.1109/RCIS.2016.7549323.
74. Martens D, Maalej W. Extracting and analyzing context information in user-support conversations on twitter. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 131–141. IEEE, 2019.
75. Han X, Li R, Li W, Ding G, Qin S. User requirements dynamic elicitation of complex products from social network service. In *2019 25th International Conference on Automation and Computing (ICAC)*, pp. 1–6. IEEE, 2019. https://doi.org/10.23919/IConAC.2019.8895140.
76. Vlas RE, Robinson WN. Two rule-based natural language strategies for requirements discovery and classification in open source software development projects. J Manag Inf Syst. 2012;28(4):11–38. https://doi.org/10.2753/MIS0742-1222280402.
77. Xiao M, Yin G, Wang T, Yang C, Chen M. Requirement acquisition from social Q&A sites. In: Liu L, Aoyama M, editors. Requirements engineering in the big data era Communications in Computer and Information Science, vol. 558. Berlin: Springer; 2015.
78. Cleland-Huang J, Dumitru H, Duan C, Castro-Herrera C. Automated support for managing feature requests in open forums. Commun ACM. 2009;52(10):68–74. https://doi.org/10.1145/1562764.1562784.
79. Morales-Ramirez I, Kifetew FM, Perini A. Analysis of online discussions in support of requirements discovery. In *International Conference on Advanced Information Systems Engineering (CAiSE)*, pp. 159–174. Springer, Cham, 2017, https://doi.org/10.1007/978-3-319-59536-8_11.
80. Khan JA, Liu L, Wen L. Requirements knowledge acquisition from online user forums. IET Softw. 2020;14(3):242–53. https://doi.org/10.1049/iet-sen.2019.0262.
81. Khan JA, Xie Y, Liu L, Wen L. Analysis of requirements-related arguments in user forums. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 63–74. IEEE, 2019.
82. Khan JA. Mining requirements arguments from user forums. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 440–445. IEEE, 2019.
83. Tizard J, Wang H, Yohannes L, Blincoe K. Can a conversation paint a picture? Mining Requirements in software forums. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 17–27. IEEE, 2019.
84. Morales-Ramirez I, Kifetew FM, Perini A. Speech-acts based analysis for requirements discovery from online discussions. Inf Syst. 2018;86:94–112. https://doi.org/10.1016/j.is.2018.08.003.
85. Merten T, Falis M, Hübner P, Quirchmayr T, Bürsner S, Paech B. Software feature request detection in issue tracking systems. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pp. 166–175. IEEE, 2016, https://doi.org/10.1109/RE.2016.8.
86. Portugal RLQ, Do Prado Leite JCS, Almentero E. Time-constrained requirements elicitation: Reusing GitHub content. In *2015 IEEE Workshop on Just-In-Time Requirements Engineering (JITRE)*, pp. 5–8. IEEE, 2015, https://doi.org/10.1109/JITRE.2015.7330171.
87. Nyamawe AS, Liu H, Niu N, Umer Q, Niu Z. Automated recommendation of software refactorings based on feature requests. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 187–198. IEEE, 2019.
88. Franch X, et al. Data-driven elicitation, assessment and documentation of quality requirements in agile software development. In *International Conference on Advanced Information Systems Engineering*, pp. 587–602. Springer, Cham, 2018.
89. Oriol M, et al. Data-driven and tool-supported elicitation of quality requirements in agile companies. Softw Qual J. 2020. https://doi.org/10.1007/s11219-020-09509-y.
90. Do QA, Chekuri SR, Bhowmik T. Automated support to capture creative requirements via requirements reuse. In *International Conference on Software and Systems Reuse*, pp. 47–63. Springer, Cham, 2019, https://doi.org/10.1007/978-3-030-22888-0_4.
91. Kang Y, Li H, Lu C, Pu B. A transfer learning algorithm for automatic requirement model generation. J Intell Fuzzy Syst. 2019;36(2):1183–91. https://doi.org/10.3233/JIFS-169892.
92. Wang C, Zhang F, Liang P, Daneva M, van Sinderen M. Can app changelogs improve requirements classification from app reviews?: An exploratory study. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 1–4. 2018, https://doi.org/10.1145/3239235.3267428.
93. Wang C, Wang T, Liang P, Daneva M, Van Sinderen M. Augmenting app reviews with app changelogs: An approach for app reviews classification. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 398–512. 2019, https://doi.org/10.18293/SEKE2019-176.
94. Johann T, Stanik C, Maalej W. SAFE: A simple approach for feature extraction from app descriptions and app reviews. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 21–30. IEEE, 2017, https://doi.org/10.1109/RE.2017.71.
95. Voet H, Altenhof M, Ellerich M, Schmitt RH, Linke B. A framework for the capture and analysis of product usage data for continuous product improvement. J Manuf Sci Eng. 2019;141(2):021010.
96. Liang W, Qian W, Wu Y, Peng X, Zhao W. Mining context-aware user requirements from crowd contributed mobile data. In *Proceedings of the 7th Asia-Pacific Symposium on Internetware*, pp. 132–140. 2015, https://doi.org/10.1145/2875913.2875933.
97. Xie H, Yang J, Chang CK, Liu L. A statistical analysis approach to predict user's changing requirements for software service evolution. J Syst Softw. 2017;132:147–64. https://doi.org/10.1016/j.jss.2017.06.071.
98. Yang J, Chang CK, Ming H. A situation-centric approach to identifying new user intentions using the mtl method. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 347–356. IEEE, 2017.
99. Wüest D, Fotrousi F, Fricker S. Combining monitoring and autonomous feedback requests to elicit actionable knowledge of system use. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 209–225. Springer, Cham, 2019.
100. Takahashi H, Nakagawa H, Tsuchiya T. Towards automatic requirements elicitation from feedback comments: Extracting requirements topics using LDA. In *Proceedings of the*

International Conference on Software Engineering and Knowledge Engineering (SEKE), pp. 489–494. 2015, https://doi.org/10.18293/SEKE2015-103.

101. Dhinakaran VT, Pulle R, Ajmeri N, Murukannaiah PK. App review analysis via active learning. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pp. 170–181. IEEE, 2018.
102. Licorish SA. Exploring the prevalence and evolution of android concerns: a community viewpoint. JSW. 2016;11(9):848–69.
103. Tizard J, Wang H, Yohannes L, Blincoe K. Can a conversation paint a picture? Mining requirements in software forums. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 17–27. IEEE, 2019, https://doi.org/10.1109/RE.2019.00014.
104. Al Kilani N, Tailakh R, Hanani A. Automatic classification of apps reviews for requirement engineering: Exploring the customers need from healthcare applications. In *2019 6th International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pp. 541–548. IEEE, 2019, https://doi.org/10.1109/SNAMS.2019.8931820.
105. Yan X, Guo J, Lan Y, Cheng X. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 1445–1456. 2013.
106. Merten T, Falis M, Hübner P, Quirchmayr T, Bürsner S, Paech B. Software feature request detection in issue tracking systems. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pp. 166–175. IEEE, 2016.
107. Barnaghi P, Wang W, Henson C, Taylor K. Semantics for the internet of things: early progress and back to the future. Int J Semant Web Inf Syst (IJSWIS). 2012;8(1):1–21. https://doi.org/10.4018/jswis.2012010101.
108. Wolpert DH. The lack of a priori distinctions between learning algorithms. Neural Comput. 1996;8(7):1341–90. https://doi.org/10.1162/neco.1996.8.7.1341.
109. Johannesson P, Perjons E. An introduction to design science. Berlin: Springer; 2014.
110. Berry DM. Evaluation of tools for hairy requirements engineering and software engineering tasks. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pp. 284–291. IEEE, 2017.
111. Dimitroff G, Georgiev G, Toloşi L, Popov B. Efficient *F* measure maximization via weighted maximum likelihood. Mach Learn. 2015;98(3):435–54. https://doi.org/10.1007/s10994-014-5439-y.