International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-024-00567-0

REVIEW

Automatic user story generation: a comprehensive systematic literature review

Carlos Alberto dos Santos[1] · **Kevin Bouchard**[1] · **Bianca Minetto Napoleão**[1]

Received: 7 January 2024 / Accepted: 15 May 2024

Abstract
User stories are the lifeblood of agile software development due to their semi-structured format and ease of implementation. However, the variety of data sources and textual formats used to document software requirements bring a challenge for software development teams. They often need to read and comprehend the client's needs from different sources and convert them into user stories manually. This process demands time, and it is also prone to errors. As an alternative to remedy this issue, there are studies concerning the automatic generation of user stories. We conducted a systematic literature review (SLR) to identify and analyze existing approaches for automatically generating user stories. We investigated which type of corpora were used for training and testing, which Natural Language Processing (NLP) or Machine Learning (ML) techniques are employed to reach this goal, and how researchers are evaluating the quality of the user stories generated. Our SLR followed established guidelines and investigated state-of-the-art research from prominent academic publishers such as ACM, IEEE Xplore, and ScienceDirect. Studies published until April 2024 were included, with a focus on those addressing the research questions proposed. Our findings indicate a critical shortage of publicly available corpora hindering advancements in this field, especially in the current era of ML. The team also found there is a broad variety of techniques being employed on this topic. Finally, the studies need to pay more attention to guidelines for evaluating user stories quality. The automatic user story generation remains in its early stages. We highlight some opportunities for contribution and discuss the direction of future works.

Keywords User story · Requirements engineering · Text generation · Systematic literature review

1 Introduction

Over the last decades, the growing complexity of software systems has pushed software developers to tackle not just technical challenges, but also human, organizational, social, and political matters [1]. Agile Software Development (ASD) [2] has emerged as a response to these challenges. At the heart of ASD lies iterative planning, enabling development teams to refine requirements incrementally, and encouraging them to engage with stakeholders. Progress is managed among team members through daily stand-up meetings, and iterative client reviews help obtain feedback about the software product in development.

This also means that ASD embraces requirements engineering activities iteratively. Hence, elicitation, specification, documentation, and validation all occur collaboratively and continuously throughout each development cycle [3]. While software requirements need to be unambiguous, testable, and agreed upon with the customer, ASD emphasizes on complete, consistent, and feasible collection of requirements [4].

In this context, ASD leverages user stories, which represent practical scenarios experienced by end users. Cohn [5] states that user stories are usually based on conversations between team members and stakeholders. They can also be written relying on requirements artifacts, such as project documents, standards, business rules, or even emails. Regardless

Kevin Bouchard and Bianca M. Napoleão have contributed equally to this work.

✉ Carlos Alberto dos Santos
carlos-alberto.dos-santos1@uqac.ca

Kevin Bouchard
kevin_bouchard@uqac.ca

Bianca Minetto Napoleão
bianca.minetto-napoleao1@uqac.ca

[1] Dép. d'informatique et de mathématique, Université du Québec à Chicoutimi, 555 Bd de l'Université, Chicoutimi, QC G7H 2B1, Canada

Published online: 03 June 2024

of the source, user stories employ a semi-structured template to express a functionality valuable for the user. The usage of a template contributes to productivity and work deliverable quality [6].

According to Stefan et al. [7], free-form and constraint textual requirements (e.g., user stories) are sufficient for agile projects acting as reminders for further discussions. The textual free-form with no constraints means a text freely written without any type of template, while the constraint textual style brings the practice of relying on a specific and simple template to express the software requirements [7]. User stories can have a defined template (e.g., "As a..., I want to..., so that...") or even be written in a free-form way to speed up the writing process. Despite its relevance, the most cited causes of failures in Requirements Elicitation (RE) are related to lack of experience and weak qualification of RE team members, lack of time, and missing completeness check of requirements [8]. The variety of data sources and textual formats used as input to RE brings a challenge for software development teams [9]. They need to read and comprehend the client's needs coming from different sources and convert them into user stories, or any other requirement specification format for development teams. This process is usually carried out manually demanding significant time and effort by the team in each software development iteration. As an alternative to remedy this issue, there are approaches [10–12] that rely on automation to support user story elicitation and specification in ASD environments.

Aiming to better understand the state of the art (SOTA) in automated support for the generation of user stories, and also motivated by emerging works toward automation in RE tasks [13–15], we performed a systematic literature review (SLR) about the automatic generation [16] of user stories based on software requirements corpora. We considered in our analysis not only approaches able to specify user stories but also elicitation techniques applied in this context. In addition, we aimed to understand how the actors evaluate the text quality of the user stories generated.

We translated our study goals into three Research Questions (RQ):

- *RQ1:* What datasets are used by researchers, and in what way, to generate user stories automatically?
- *RQ2:* Which techniques are used to specify user stories automatically?
- *RQ3:* How user story quality is evaluated?

As the main contributions of this study, we highlight (i) a list of user stories corpora available for usage in different studies related to this topic; (ii) a discussion of different techniques employed to automatically generate user stories; (iii) an analysis of the different guidelines to evaluate user stories quality; (iv) a discussion about the possible future work on this topic, analyzing what was proposed by the authors of the primary studies.

This paper is organized as follows: Section 2 presents base concepts about agile requirements engineering and user stories definition. Section 3 discusses some essential related work of our topic. In Sect. 4, we detail our study design by describing the SLR protocol. We present our results by answering the proposed research questions in Sect. 5. In Sect. 6, we discuss our results by exploring each research question's perspectives and research opportunities. Section 6.5 describes the limitations of our study and their mitigation strategies. Finally, Sect. 7 concludes our study.

2 Background

Story cards, introduced by Kent Beck in the Extreme Programming methodology [17], use concise sentences to outline system behavior, stimulating team discussions. They were broken down into tasks for the development team to tackle. Mike Cohn, building on Beck's idea, refined the concept [5], emphasizing that they should contain a brief description of user or customer-focused functionality. However, the core aspect remained: the ongoing dialogue between stakeholders and developers regarding the story itself. Cohn also provided guidelines for crafting effective stories to guide the software development process.

User stories can be authored by any team member or stakeholder at various project stages. They typically emerge from conversations between team members and stakeholders or from various requirements sources such as project documents, standards, business rules, or emails [5]. These user stories follow a semi-structured template to express valuable user functionality, and employing this template enhances productivity and the quality of deliverables [6]. Each user story aims to define an *actor*, an *goal*, and the *benefit*, or business value delivered by that action. For instance, the actor (or role) could be the end-user, and a goal could be an operation made by the actor at the system or business level, such as printing a document, scanning a barcode or returning a product to the warehouse. Finally, the benefit represents what the actor will accomplish through that goal.

As a <***role***>, *I want* <***goal***>, *so that* <***benefit***>.

It brings the practice of focusing on the client's perspective about the system and the business. As the user story template is simple by itself, the writer cannot add too many details to the story. So, rather than writing all details as stories, the better approach is to encourage communication between the development team and the customer. There is no problem in writing down some notes on the story cards, but the conversation is the goal when working with user stories [5]. The communication motivated by the user stories can improve

the clarity and understanding of the software requirements. This is an example of a user story extracted and adapted from a project[1] of an online platform to support waste recycling:

> *As a* ***user****, I want* ***to be able to get a list of nearby recycling facilities****, so that* ***I can determine which ones I should consider****.*

Lucassen *et. al* [18] evaluated the use and effectiveness of user stories in practice. They state that the user story is the smallest representation of requirements used by software developers to build new features daily. The use of a template to express user stories is a standard industry practice. Different templates can be used to assist development teams in writing user stories. The most used user story template is *Connextra* (example above), which originated with agile coach *Rachel Davies* in the early 2000s [19]. Only 15% of the respondents do not use templates [18]. Those who use templates agree that the user story representation supported by templates contributes to productivity and work deliverable quality [6, 18].

User stories play an important role in helping development teams build the right product [18]. As they are smaller pieces of requirements, the stakeholders are pushed to depict requirements with more details to development teams. The more details they have the easier it is to code the right product. It also reinforces communication and brings the development team and stakeholders to the stage in the early phases of the project. In fact, it has been shown that stakeholders like to work with user stories [18]. This practice helps development teams to prevent later defects in the software.

On the other hand, Pokharel et al. [20] produced another study of user stories in practice where they found that not all agile practitioners use user stories. Only 45% of their respondents are committed to applying user stories in agile software development, and only 15% of those have sound knowledge about this practice. In addition, they state that backend developers are more prone to not use user stories, as opposed to full-stack developers who take more advantage of this practice. In summary, they conclude that development experience alone is not enough to gather client needs. Effective training on user stories and agile principles is required to play agile practices in organizations.

3 Related work

To the best of our knowledge, no prior secondary studies have specifically focused on the automatic generation of user stories. However, we found some relevant secondary studies related to user story practice to guide our research. These studies, detailed below, offer valuable insights

[1] https://data.mendeley.com/datasets/7zbk8zsd8y/1.

3.1 Reviews

Amna et al. [21] conducted a systematic literature mapping of user story research. Their dataset was composed of 186 unique peer-reviewed papers, published in 2001–2021. They found that research on user stories practice is, in general, recent (2015–2021), representing 78% of the total of papers selected. Next, they noted that about 80% of the papers are solution-oriented, the other 20% being related to problem investigation or observation. Most proposed solutions are based on algorithms to solve well-defined problems related to system design issues during requirements analysis and negotiation activities, i.e., algorithms to estimate project variables, prioritization, and project schedule optimization. For less well-defined problems, such as ambiguity in user stories, they [21] found that the proposed solutions involve the use of conceptual models or software models, based on different artifacts to support the user story writing. While the models help to organize the relationship and dependencies between the stories, the artifacts, such as ontologies and glossaries, help the writers to produce better user stories. Otherwise, they conclude that there is a lack of tested solutions to mitigate the formulation of ambiguous user stories in RE.

Raharjana et al. [22] presented a systematic review of Natural Language Processing (NLP) applied to user stories. Their dataset was composed of 38 unique peer-reviewed papers, published between 2009 and 2020. They found that the authors usually rely on Semantic Role Labeling (SRL) to identify the key abstractions of user stories (who, what, and why) and generate models/artifacts. Respectively, it contributes to a better understanding of the format of user stories and also accelerates the development lifecycle. Next, they mapped which NLP techniques are usually used in this context. They listed: *preprocessing, POS tag, named-entity recognition, vector space model, dependency, syntactic parse tree and bag-of-words (BoW)*. To finish, they also listed the challenges of using NLP in user story research: dataset heterogeneity, manual data tagging, context domain-dependent, human intervention and low recall and precision over the studies selected.

On a similar topic, Cheligeer et al. [23] published a literature review on Machine Learning (ML) applied to requirements elicitation. RE or requirements discovery is the process of surfacing candidate requirements from different sources [4]. Well-executed elicitation helps minimize software requirements incompleteness [4]. Cheligeer et al. [23] selected 86 articles for this study. They identified 15 different ML-based requirement elicitation tasks and 12 different data sources for building a data-driven model. In addition, they analyzed and categorized the techniques for constructing ML-based requirement elicitation methods into five parts. They found that BoW language models and handcrafted features are frequently used as a technique. But also,

an increasing trend toward using embedding features has been observed. The most used ML algorithms to automatically elicit requirements are Naive Bayes, support vector machines, decision trees, and neural networks. For evaluation, Precision, Recall, and F1 score are the most prevalent evaluation metrics applied to assess model performance. In the realm of natural language processing tools, NLTK [24] and CoreNLP [25] are widely recognized as the most commonly employed, while for ML training, Weka [26] and Scikit-learn [27] are the most popular choices. The authors conclude that despite some studies lacking evaluation and concrete evidence, ML can theoretically and practically support requirements activities.

Also, Zhao et al. [28] contributed to this area by mapping the state of the art in natural language processing for requirements engineering (NLP4RE). In summary, they concluded that the past years were successful in providing technical solutions to cover the RE lifecycle, and now it is time to better evaluate those solutions with empirical assessment in real scenarios in order to lead to more powerful tools for RE. In addition, they found that although user stories, interview scripts, domain documents, use cases and models are primarily used in ASD practice, they still need more attention from researchers.

Although the related literature studies presented in this section offered valuable insights into user story practices (Table 1), there were no studies related to the automatic generation of user stories combining elicitation and specification processes for this goal. Integrating such research is of paramount importance, as it will serve to both strengthen and provide clear guidance to ongoing research pursuits on this topic.

3.2 User story quality

One of the goals of this SLR is to identify how the studies evaluate the writing quality of the user stories (RQ3). Much has already been achieved in terms of identifying incorrectness in software requirements statements. In the following paragraphs, we briefly discuss the most relevant works related to the quality evaluation of user stories.

Amna [29] et al. prepared a systematic review of user story ambiguity. This work makes an important contribution by defining and analyzing the different levels of ambiguity in user stories (i.e., lexical, syntactic, semantic, pragmatic). Their dataset brings 36 studies published from 2001 to 2020 committed to evaluating ambiguity in user stories. They found three main gaps in research on this topic. First, there is a need for more research on human behaviors and cognitive factors that cause ambiguity. Second, the studies found rarely consider a set of user stories when analyzing ambiguity. They only focus on unique statements, but a set of user stories is prone to inconsistency, that is a kind of ambiguity. Third, the solutions proposed do not consider ambiguity in different linguistic levels as presented by the authors. The studies need to rely more on linguistic theory to work with ambiguity in user stories.

In terms of regulations, ISO standards on software requirements have helped to set the basis of the knowledge about quality and good practices in the requirements specification. Various research efforts exist to improve the quality of requirements statements. Some of them are using methods purely focused on NLP [30, 31], others trying to solve requirements problems through the employment of ML algorithms [32, 33]. Many of these works bring their understanding of categories of requirements problems, and they propose different frameworks to identify these issues on requirements statements.

Femmer *et. al* [34] worked to standardize the identification of poor quality in requirements statements by introducing the concept of *requirement (bad) smells*. They developed this theory based on ISO 29148 requirements engineering standard [35], as it provides a list of *requirements language criteria*, which should help to choose a proper language for requirements specifications. The authors developed a tool to assist the detection of Requirements Smells by using part-of-speech (POS) tagging, morphological analysis and dictionaries. They performed a series of experiments in different software requirements corpora, including user stories, and concluded that Requirements Smells can point relevant defects in different forms of requirements, domains, and other methodologies followed, including agile. Other authors have been using Femmer's papers as a baseline for studies related to bad smell identification in requirements statements [36, 37]. The requirement smells concept is relevant as it represents a big picture in terms of quality faults in requirements artifacts, also making it possible to apply them in agile environments.

On the other hand, other authors dedicate their efforts strictly to evaluating the quality of user stories. It is the case of Lucassen et al. [38] on proposing the Quality User Story (QUS) framework and the Automatic Quality User Story Artisan (AQUSA) tool to evaluate the quality of user stories in agile environments. QUS framework defines 13 language criteria for user story quality divided into three categories: syntactic, semantic and pragmatic. QUS also defines quality criteria for a set of user stories as a goal to verify the quality of a complete project specification.

Lucassen et al. [38] developed the AQUSA tool as a goal to support the framework employment. It can automatically identify the language criteria defined by the framework and point out the defects that need user correction. This tool was coded in Python using the Flask microframework. It also relies on Stanford CoreNLP [25] and the Natural Language ToolKit (NLTK) [24] tools, using a set of algorithms based on language heuristics. It is built as an API able to connect to

Table 1 Summary of related studies

Title	Goal	Concerns in research questions
User Stories and Natural Language Processing: A Systematic Literature Review (Raharjana, 2021 [22])	Capture the current state-of-the-art of NLP research on user stories	Uses of NLP for user stories Available NLP approaches for user stories Challenges of using NLPs in user story
Systematic Literature Mapping of User Story Research (Amna, 2022 [21])	Investigate user stories research to identify problems and solutions proposed by researchers, evaluate the level of maturity of the area and research gaps existent	Identify research areas related to user story technique Diagnose the type of problems in the area Analyze the kind of outcomes produced Evaluate how the research is conducted Identify the type of publications
Machine learning in requirements elicitation: a literature review (Cheligeer, 2022 [23])	Summarizes and analyzes studies that incorporate ML and NLP into demand requirements elicitation	Analyze requirements elicitation activities supported by ML Identify requirements data sources Identify technologies, algorithms, and tools to build ML-based elicitation Framework to build ML-based solutions for elicitation
Natural Language Processing for Requirements Engineering: A Systematic Mapping Study (Zhao, 2021 [28])	Map the state-of-the-art in natural language processing applied on requirements engineering	Analyze publication data in NLP4RE Analyze the state of the empirical research in NLP4RE Identify the focus of research on NLP4RE Inspect the tool development in NLP4RE Identify NLP techniques used
Ambiguity in user stories: A Systematic Literature review (Amna, 2022 [29])	Review the studies that investigate or develop solutions for problems related to ambiguity in user stories	How researchers define ambiguity problems? Identify solutions for ambiguity in user stories Analyze evidence of the effectiveness of the identified solutions

The reference [29] is discussed in Sect. 3.2

other tools, such as Jira or even Excel. The authors evaluated their tool using 18 sets of real-world user stories. AQUSA was able to identify defects in 56% of these user stories. They concluded that for some syntactic quality criteria, it is possible to achieve results close to the Perfect Recall Condition. It demonstrates how the AQUSA tool and QUS framework can be used in industry environments to support user story quality enhancement.

The INVEST [39] framework is a well-known tool used to evaluate user stories. It states that good user stories contain the following set of characteristics: Independent, Negotiable, Valuable, Estimable, Small and Testable. They define each one of these characteristics in a very simple way as a goal to facilitate its usage. Lucassen et al. [18] prepared a survey to understand how practitioners use and perceive user stories in real-world scenarios. They found that 33% of the respondents apply some guidance for evaluating user stories quality, being that 23.5% of respondents apply the INVEST framework in their projects. Though other 39.5% of the public does not validate the user stories using any type of quality guidelines.

Lucassen et al. [18] stated in their research that the usage of quality guidelines in user stories improves productivity and the template adoption further improves work deliverable quality. This statement is one of the reasons why we are committed to investigating how the researchers are evaluating the quality of the user stories automatically generated. There are abounding approaches that can be used to generate text in the field of artificial intelligence, and their adoption depends on the context of their employment. The results produced by text generation algorithms shall be evaluated through qualitative measures in order to ensure they are aligned with the needs of the environment.

4 Research design

As already mentioned, this study aims to provide a better understanding of the current state of the art on automated support for the generation of user stories exploring literature-available datasets to generate user stories, techniques used to generate user stories as well as comprehend how user story quality is evaluated by researchers. We followed Kitchenham et al.'s guidelines for SLR [40] to identify and summarize available literature based on outlined RQs. In this Section, we present details about the RQs and report the SLR protocol steps [40] performed in our study. In Table 2, we detail the rationale behind the consideration of each RQ.

In Sect. 4.1, we describe our search strategy and selection criteria adopted to systematically retrieve and select published research on user story generation. Section 4.2 relates the quality criteria analysis performed to evaluate the quality of the selected studies. Finally in Sect. 4.3, we explain how we extracted and synthesized data to answer the proposed RQs.

Table 2 Research questions and their rationales

#	Research question	Rationale
RQ1	What datasets are used by researchers, and in what way, to generate user stories automatically?	This research question provides an understanding of the existence of specific dataset sources used by researchers to generate user stories automatically
RQ2	Which techniques are used to specify user stories automatically?	This question looks for techniques and details on their application to specify user stories automatically
RQ3	How user story quality is evaluated?	Provides an overview of how user stories have been evaluated by researchers regarding their quality

4.1 Search strategy and study selection

We adopted a hybrid (two-stage) search strategy combining database search with snowballing [41]. First, we performed a database search [40] and then a snowballing search [42]. A database search consists of developing and running a search string in a digital library or digital library [40]. A snowballing search analyses the references (backward) and citations (forward) of selected studies aiming to detect other relevant studies [42]. According to Mourão *et al.* [41], the combination of database search from the *Scopus*[2] digital library with forward and backward snowballing represents an appropriate alternative to be adopted as a search strategy for SLRs.

To perform the database search, we identified suitable keywords. The keywords definition was initially based on the authors' knowledge and past experiences [43]. Additionally, as suggested by Kitchenham et al. [40], we reviewed our RQs and identified important concepts and terms used in the RQs and related to the main topic of interest in order to generate our group of domain keywords. Next, we tested the capacity of the multiple combinations of the domain keywords to retrieve a set of studies known by the authors (control group) in *Scopus* digital library. As recommended by Kitchenham et al. [40], the study control group was used to calibrate our search string. In the following, we present our full search string.

```
  (("user stor*") AND (processing OR
automat* OR extract* OR "text predict

*" OR generat* OR "natural language" OR
```

[2] https://www.scopus.com.

```
nlp
OR "machine learning" OR ml ))
```

We ran our final search string on *Scopus* digital library in April 2024 in three metadata fields: title, abstract and keywords. The search string was also adapted to meet specific search criteria (e.g. syntax) of the digital library. Besides the recommendation of Mourão et al. [41], we chose Scopus because it indexes studies of several international publishers, including *Springer*, *Wiley-Blackwell*, *Elsevier*, *IEEE Xplore* and *ACM Digital Library*. As a result of our database search, 637 studies were retrieved. Next, selection criteria were defined for deciding which documents retrieved were relevant. The selection criteria are organized into three Inclusion Criteria (IC) and Exclusion Criteria (EC):

- **IC1:** The study must present or mention the automated generation of use story adopting NLP and/or ML techniques; *AND*
- **IC2:** The study must be peer-reviewed.
- **EC1:** The study is not a journal or conference paper; *OR*
- **EC2:** The study is not written in English; *OR*
- **EC3:** The study is an older version of another study already considered.

We first performed cleaning in our dataset of 637 retrieved documents by removing conference announcements and documents that were out of the scope of the computer science domain. This stage resulted in 524 documents. Next, we applied the IC and EC based on the analysis of the title, abstract and keywords. Given the high number of documents to be analyzed, these criteria were manually applied by the corresponding author only. However, any doubt about and inclusion or exclusion of a study was discussed with other authors through consensus meetings. A total of 16 studies were selected as candidate studies at this stage. Lastly, the IC and EC criteria were applied based on the full-text analysis of the candidate studies. 5 out of 16 studies were excluded, resulting in 11 included studies during the database search analysis.

It is worth mentioning that during the analysis of the title, abstract and keywords of the studies retrieved from Scopus, 4 secondary studies [21–23, 29] (SLRs and systematic mappings—a kind of lightweight SLR that aims to survey the available knowledge about a research topic [40]) were identified (listed in Sect. 3.1). Since they are not primary studies (e.g. case studies, empirical studies, surveys, etc.), we opted to consider them separately as related work. They are described in Sect. 3.

Following the second stage of the hybrid search strategy [41], we perform forward and backward snowballing aiming to avoid missing relevant studies. We followed the guidelines proposed by Wohlin [42] to perform the snowballing search. We used the 11 included studies from our database search process + 4 secondary studies considered as related work as our "seed set" (starting set) for performing two iterations of the snowballing technique. We performed the forward and backward snowballing techniques in parallel (i.e., backward then forward, and so on). The studies' citations were extracted with the support of digital libraries, such as *Google Scholar*[3] and *Scopus*. In each snowballing iteration, we applied IC and EC criteria first on the title, abstract and keywords, and next on the full text. We performed two backward and forward snowballing iterations, stopping their execution at the last iteration because no more relevant study was detected. One author performed the snowballing technique, and when doubts about the inclusion or exclusion of a paper were raised, it was discussed with other authors in meetings. As a result of both snowballing techniques, we added 6 (5 first rounds and 1 second round) more relevant studies to our set of included studies, totaling a final set of 17 included studies. The final list of the included studies is

Table 3 Quality assessment questions [44]

#	Questions	Rationale
	Proposed-of-solution research	
QA1	Is the problem to be solved by the technique clearly explained?	The scope of the problem must be delimited in the paper. Assess the level of understanding of the problem by the authors.
QA2	Is the technique novel, or is the application of the techniques to this kind of problem novel?	Level of novelty regarding the technique or solution presented in the paper in comparison with related work discussed by the paper's authors or existing literature until the paper's publication time.
QA3	Is the technique sufficiently well described so that the author or others can validate it in later research?	Perception at the level of replicability and reproducibility of the presented technique/solution.
QA4	Is the broader relevance of this novel technique argued?	The significance and potential impact of the technique/solution in a general view of the domain under investigation.
QA5	Is there sufficient discussion of related work?	The mention and the discussion of strengths and/or limitations of previous approaches and how the current paper is motivated by them or addresses these limitations.

[3] https://scholar.google.com.

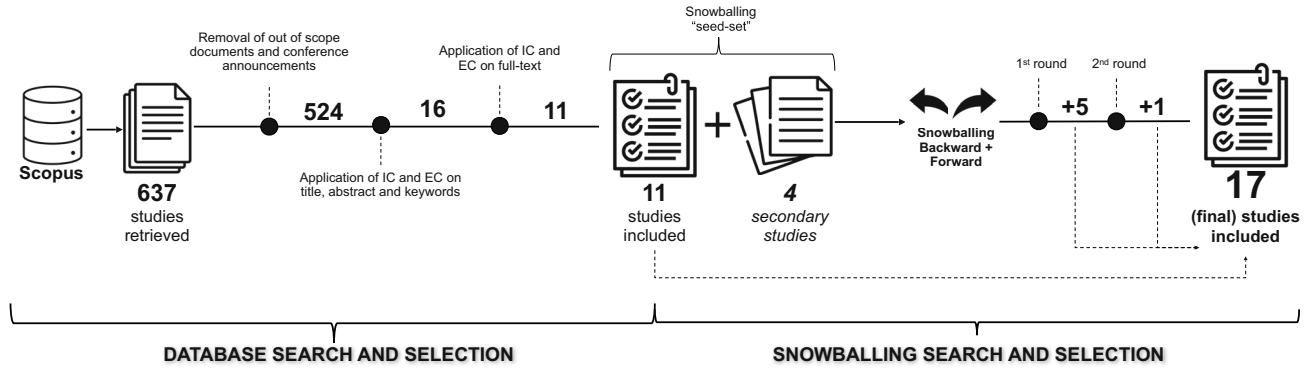

Fig. 1 Search strategy and selection process

presented in Table 6. Figure 1 illustrates the whole search and selection process and its findings.

4.2 Quality assessment

We followed the guidelines proposed by Wieringa *et. al* [44] to assess the quality of the selected primary studies. This study recommends an evaluation criteria for different published studies in the RE field. We adopted the five Quality Assessment (QA) questions recommended for studies that present solutions for RE problems (Table 3). For each question we assigned a score, being (1) if the primary study explicitly addressed the QA question; (0.5) points if the question is partially addressed and (0) for the QA questions that do not present related evidence in the study.

In order to facilitate the comprehension of our analysis, we presented the rationale used by the authors during the analysis of the papers to answer each QA question. In Table 4, we present the final sum of the scores for each primary study selected. The QA was carried out by the corresponding author and by the author who has almost 10 years of experience in conducting SLRs. The scores presented in Table 4 resulted from consensus between the authors.

The objective of QA is to ensure that the studies included in the SLR present an acceptable level of quality.

It is worth mentioning that the adopted criteria were simply guidelines rather than quantitative evaluations with standardized values. The evaluation of the papers was based strictly on the interpretation of the authors considering the rationale behind each question. Amna et al. [29] also followed the QA procedures proposed by Wieringa et al. [44]. We present our QA results using the same format and criteria used for Amna [29]. As illustrated in Table 4, all 17 studies included by IC and EC were retained as included studies after the QA.

4.3 Data extraction and analysis

To extract all relevant data to answer our RQs, we created a data extraction form based on our RQ goals. We used this predefined data extraction form as specified in Table 5 to organize data extracted from the different primary studies. The data were extracted by the corresponding author and samples of the data were reviewed by a second author in order to validate the data extraction phase.

The extracted data is publicly available online.[4] In this document, there is the extracted data for each primary study selected in our SLR. Each tab of the data extraction document refers to the data extracted from one paper following the form structure proposed in Table 5.

The data synthesis was performed through a combination of content analysis from the extracted data categorizing the findings into broad thematic categories [59] as well as a narrative synthesis [60]. The extracted information was interpreted and analyzed considering the authors' knowledge and experience in requirement elicitation and analysis. In the case of disagreements, consensus was established through discussion. The results of our analysis are described in Sect. 5 by answering the proposed RQs.

5 Results

In this Section, we present the comprehensive findings of our SLR. The 17 selected papers are listed in Table 6, providing a concise overview of the specific studies selected in our study.

5.1 Summary of studies

Of the 17 included primary studies in our SLR, 13 of them (76%) were published in conferences. The other four studies were published in journals (24%). We considered short

[4] https://doi.org/10.5281/zenodo.8395878.

Table 4 Quality assessment scores

Study	Year	QA1	QA2	QA3	QA4	QA5	Score	Qual. (%)
Thamrongchote et al. [45]	2016	0.5	1.0	0.5	0.5	0.0	2.5	50
Rodeghero et al. [10]	2017	0.5	1.0	1.0	1.0	1.0	4.5	90
Murtazina et al. [46]	2018	0.5	0.5	0.5	0.5	0.5	2.5	50
Santos et al. [47]	2018	0.5	0.5	1.0	0.0	0.5	2.5	50
Raharjana et al. [11]	2019	1.0	1.0	0.5	0.5	0.5	3.5	70
Li et al. [48]	2019	1.0	1.0	0.5	0.5	1.0	4.0	80
Veitia et al. [12]	2020	1.0	1.0	0.5	0.5	1.0	4.0	80
Resketi et al. [49]	2020	1.0	1.0	1.0	1.0	1.0	5.0	100
Henriksson et al. [50]	2020	1.0	1.0	1.0	1.0	1.0	5.0	100
Panichella et al. [51]	2020	1.0	0.5	0.5	0.0	0.5	2.5	50
Nistala et al. [52]	2022	1.0	1.0	1.0	1.0	1.0	5.0	100
Kumar et al. [53]	2022	1.0	1.0	0.5	0.0	0.5	3.0	60
Lam et al. [54]	2022	1.0	1.0	0.5	0.0	1.0	3.5	70
Dwitam et al. [55]	2022	0.5	1.0	0.5	0.5	0.0	2.5	50
Heng et al. [56]	2023	0.5	0.5	0.5	1.0	1.0	3.5	70
Mateus et al. [57]	2023	1.0	1.0	1.0	0.5	0.5	4.0	80
Siahaan et al. [58]	2023	1.0	0.5	1.0	1.0	1.0	4.5	90

Table 5 Data extraction form

#	Study data	Description	Relevant RQ
1	Identifier	Unique ID for the study	Study overview
2	Title		Study overview
3	Authors		Study overview
4	Year		Study overview
5	Article source		Study overview
6	Type of article	Journal, conference, workshop, book chapter	Study overview
7	1st author country		Study overview
8	Application context	Industrial, academic	Study overview
9	Date of data extraction		Study overview
10	Research Goal		Study overview
11	Data	Datasets used to automatically generate user stories	RQ1
12	Data	Is the dataset used publicly available?	RQ1
13	Data	How the datasets are used to generate user stories?	RQ1
14	Techniques	Is machine learning used?	RQ2
15	Techniques	Is natural language processing used?	RQ2
16	Techniques	Which techniques are used to generate user stories?	RQ2
17	Validation	How user story quality is evaluated?	RQ3
18	Validation	Some specific user story framework is used?	RQ3
19	Challenge and limitation	What challenges and limitations did the study acknowledge?	Study overview
20	Future work	What future work did the authors suggest?	Study overview

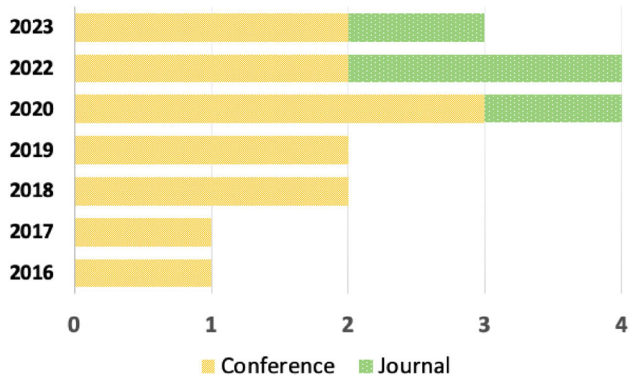

Fig. 2 The number of collected conference and journal papers until April 2024

papers in our datasets because they bring essential contributions in terms of available approaches though not all present rigorous validation steps. Of the total of studies selected, eight (47%) are full papers and the rest (53%) are short papers. The documents were distributed in different venues, indicating no one venue's preference for publication on this topic. The graph illustrated in Fig. 2 presents the number of journal and conference papers retrieved by year of publication.

The predominance of short papers and the lack of well-defined publication venues strongly suggest that this area is still in its early stages of development. In fact, it still needs more empirical evaluation of the approaches proposed as a way to validate if it remains an open and promising topic in RE. As user stories are short and simple, development teams are used to write them manually. The advancements in machine learning, language models and the conjunction of different NLP techniques can bring opportunities to automate the writing of user stories in software development projects.

5.2 (RQ1) What datasets are used by researchers, and in what way, to generate user stories automatically?

The selection of valuable datasets is primary for the success of approaches that involve ML. It also makes it possible for other researchers to reach the same results by reproducing the experiment using the same dataset or comparing it with different approaches. In this Section, we identified the datasets used to specify user stories in the selected primary studies. We also investigated how they were used by the researchers to achieve that goal.

We identified 16 different software requirements datasets in our primary studies. However, only eight of them are publicly available and the access link can be found in Table 6. No attempt was made to contact them to gain access to the datasets, so we only share the access links provided by the authors along with their papers. The public datasets can be reused to validate different approaches involving the generation and validation of user stories in different contexts. The rather low percentage of public datasets among the studies may be partially explained by the nature of software engineering; companies tend to keep their development process tightly closed to the public [61]. Note that we also included two lexical dictionaries in our list, Wordnet [62] and Bloomsoft [63], used by Siahaan [58] to create a dictionary of terms present in user stories statements.

Rodeghero et al. [10] used transcripts of spoken conversations during requirements gathering meetings to identify user story information. This paper does not present an approach to generating user stories but only identifies user story data in conversation transcriptions. The dataset[5] used in this research was manually annotated to extract these parts from the text. Then, they trained an ML classifier to automatically recognize function and rationale information in the corpus.

Also, Veitia et al. [12] contribute by sharing their two software requirements datasets[6][7] used to identify user stories in software's issues records. As in the previous paper, the authors do not generate user stories, but they use natural language processing and machine learning models to identify user story information in the software issue tracker corpus.

One of them[8] was also used by Kumar et al. [53] in their work about user story splitting. The authors presented an approach to split complex user stories into less complex ones. For example, user stories representing CRUD operations (Create, Read, Update and Delete) could be split into four user stories (create, read, update, and delete), allowing the development team to better address the coding and testing steps. The authors automatically identified the complex user stories using a Python script based on linguistic heuristics.

Then, Resketi et al. [49] shared their dataset used to summarize user stories automatically. This dataset was derived from a project aimed at facilitating the generation of metadata for digital collections at Duke University libraries while concurrently establishing a digital repository for digitized materials.

Heng et al. [56] shared the dataset containing user stories and Behavior-driven Development scenarios (BDD). They used an ontology to organize data and build user stories. They believe that the use of an ontology can produce higher-quality user stories.

In addition to the lexical dictionaries Wordnet [62] and Bloomsoft [63] used by Siahaan et al. [58], they also used PURE, the Public Requirements Documents Dataset to sup-

[5] https://groups.inf.ed.ac.uk/ami/corpus/.

[6] https://data.mendeley.com/datasets/7zbk8zsd8y/1.

[7] https://connectopensource.atlassian.net/issues/?jql=orderbycreatedDESC&startIndex=50.

[8] https://data.mendeley.com/datasets/7zbk8zsd8y/1.

Table 6 Primary studies and the datasets used to automatically identify and/or generate user stories

Study	Year	Type	Size	Dataset	Available?
Thamrongchote et al. [45]	2016	Conference	Short	Historical data	No
Rodeghero et al. [10]	2017	Conference	Full	Transcripts	Yes
Murtazina et al. [46]	2018	Conference	Short	–	No
Santos et al. [47]	2018	Conference	Short	Use Cases UH4SP	No
Raharjana et al. [11]	2019	Conference	Short	Online News Dataset	No
Li et al. [48]	2019	Conference	Short	–	No
Veitia et al. [12]	2020	Conference	Short	Issue Tracker CONNECT	Yes
Veitia et al. [12]	2020	Conference	Short	Requirements datasets	Yes
Resketi et al. [49]	2020	Journal	Full	Industry dataset	No
Resketi et al. [49]	2020	Journal	Full	Trident project	Yes
Henriksson et al. [50]	2020	Conference	Full	–	No
Panichella et al. [51]	2020	Conference	Short	–	No
Nistala et al. [52]	2022	Journal	Full	Industrial dataset	No
Kumar et al. [53]	2022	Conference	Short	Requirements datasets	Yes
Lam et al. [54]	2022	Conference	Short	Chatbot conversation	No
Dwitam et al. [55]	2022	Journal	Full	Chatbot conversation	No
Heng et al. [56]	2023	Conference	Full	US/BDD scenarios	Yes
Mateus et al. [57]	2023	Conference	Full	–	No
Siahaan et al. [58]	2023	Journal	Full	PURE	Yes
Siahaan et al. [58]	2023	Journal	Full	Bloomsoft	Yes
Siahaan et al. [58]	2023	Journal	Full	Wordnet	Yes
Siahaan et al. [58]	2023	Journal	Full	Online News Dataset v2	No

port the creation of the user stories terms dictionary. This is a well-known dataset already used by other authors to propose diverse techniques in RE.

On the other hand, eight other datasets used by the authors are not publicly available. It happens due to industrial rights protection or even because the authors might overlooked sharing their datasets. For example, Thamrongchote et al. [45] based their approach validation on the historical data of a project, but did not publish the dataset. Notably, they used this data to populate an ontology, which is essentially a structured representation of knowledge within a specific domain [64]. In this case, the ontology contained terms and relationships related to user stories, aimed at facilitating their creation and reuse.

Raharjana et al. [11] and Siahaan [58] et al. created online news datasets; however, they did not share them with the readers. The authors used semantic role labeling to extract the aspects of *who, what and why* from online news datasets to automatically create user stories using different natural language processing techniques. According to the authors, the usage of specified online news corpus in RE can contribute to a better domain knowledge understanding. We found these studies particularly interesting because of the nature of the datasets used and how the proposed techniques were able to collect data from the news text.

Mateus et al. [57] proposed user story generation by applying transformation patterns to BPMN graphs. They present a well-described example of transformation in their paper though they do not use any specific dataset to perform a more complete evaluation of the proposed technique.

It also draws attention to the other three short papers that do not use any dataset to validate their approach [46, 48, 51], as they are only concerned with explaining an idea or approach without deep data validation. Also, Henriksson et al. [50] did not use any datasets when proposing their approach for automated requirements elicitation, though the paper presents a transparent metadata model to collect software requirements data.

5.3 (RQ2) Which techniques are used to specify user stories automatically?

For this study, we are considering using NLP and ML techniques for automatic user story identification and specification. Some papers also rely on manual tasks, such as natural language annotation and transformation rules. The goal of this research question is to identify the most used techniques and how they collaborate with the approaches proposed. Usually, a paper is going to combine different techniques to reach a goal. Table 7 lists an overview of the most important techniques or tools presented in the papers.

Table 7 Main techniques used for each primary study to identify and generate user stories automatically

Study	NLP	ML	Main technique
Thamrongchote et al. [45]	Yes	No	Ontology
Rodeghero et al. [10]	No	Yes	ML classification
Murtazina et al. [46]	Yes	No	Ontology
Santos et al. [47]	Yes	No	Language heuristics
Raharjana et al. [11]	Yes	No	NER
Li et al. [48]	Yes	No	Graph theory
Veitia et al. [12]	Yes	Yes	Neural Networks
Resketi et al. [49]	Yes	No	Bag of words
Henriksson et al. [50]	Yes	Yes	NER
Panichella et al. [51]	No	Yes	ML classification
Nistala et al. [52]	Yes	No	Text Parser
Kumar et al. [53]	Yes	Yes	ML clustering
Lam et al. [54]	Yes	No	Chatbot
Dwitam et al. [55]	Yes	No	Chatbot
Heng et al. [56]	Yes	No	Ontology
Mateus et al. [57]	Yes	No	Language heuristics
Siahaan et al. [58]	Yes	No	POS tagging

Ontologies were used by Thamrongchote et al. [45], Murtazina et al. [46] and Heng et al. [56] with different goals. The first paper uses ontologies to organize user story data for reuse in different projects. By doing that, creating new artifacts simpler and faster is possible. So, the second paper uses ontologies as a way to help the user evaluate the written quality based on ontology rules. Finally, the third one uses the dataset presented to populate an ontology and validate it. As a second step of this research, the authors intend to create a prototype tool to assist the writing of user stories.

Data collection plays an important role in the generation of user stories. We believe that the reasoning behind data collection and organization is part of the scope of this study. In this case, the studies [45, 46, 56] take advantage of the user story template to organize data into ontologies serving as a database for diverse employments. It reinforces the simplicity of user story structure and how they are straightforward to employ.

Lam et al. [54] and Dwitam et al. [55] used chatbot applications to support the elicitation step on requirements engineering. Firstly, Lam et al. [54] used the IBM Watson chatbot system in a web application to gather clients' requirements and suggest user stories. They trained the chatbot in that way and presented all the aspects involved in this training. However, this solution is presented in the format of a short paper without a deeper analysis of its effectiveness. Also, the technology dependence might make it harder to replicate the study in other contexts.

Secondly, Dwitam et al. [55] also used a chatbot for the same purpose. They relied on AIML (*Artificial Intelligence Markup Language*) to train the chatbot and made more use of natural language processing as a pre-processing step to deal with the Indonesian language. Both projects [54, 55] bring important contributions in terms of using Artificial Intelligence (AI) in RE, though the solutions proposed are corpus-dependent and need different chatbot training to adapt them to different contexts.

Raharjana et al. [11] and Henriksson et al. [50] made use of NER (*Named Entity Recognition*) technique to deal with user stories. Raharjana et al. [11] also explored the usage of Part-of-speech (POS) tagging combined with language heuristic. The processing steps were divided into role clustering, weighing and filtering to extract user story data from domain online news corpora. They proposed the composition of user stories by using language transformation rules.

Differently, Henriksson et al. [50] propose a data model able to organize the RE process coming from different data sources. The data model presented can also store user stories converted from input data. To do that, they recommend a data extraction step based on classification algorithms, and sentiment analysis, in addition to NER. This is a theoretical paper not committed to presenting technical background details, and therefore, it is difficult to assess the feasibility of the approach in a realistic software development context.

Siahaan et al. [58] prepared a study similar to [11]. They also used NLP techniques to identify user story data in online news datasets and compose user story statements using language heuristics. The techniques employed by the authors include part-o-speech (POS) chunking, named-entity recognition (NER) and dependency parsing. However, they relied on lexical dictionaries (WordNet and BloomSoft) to support term identification related to software engineering.

Rodeghero et al. [10] relied on machine learning classification to recognize user story information in team conversations. They first prepared a qualitative study to test the hypothesis that conversations between developers and customers contain role, function, and rationale information for user stories. Secondly, they performed a quantitative study to determine the degree to which an existing classification algorithm can be trained to recognize this information in conversations. At the time of the study, they concluded that a Logistic Regression model was more prone to identify user story information in conversation corpora. They compared their results with the annotated corpus and they also used the standard machine learning performance metrics, such as Precision, Recall and others for evaluation. Nevertheless, since then, more advanced machine learning methods have emerged[9] and with state-of-the-art methods, the results would probably improve drastically.

[9] https://paperswithcode.com/task/text-classification.

Santos et al. [47] presented a theoretical approach using language heuristics capable of deriving logical architectures as a goal to define the initial requirements, in the form of user stories. They performed a controlled experiment deriving a module of architecture into a set of themes, epics, use cases and user stories. The authors believe they need to extend the number of transformation rules to cover other different types of requirements.

Mateus et al. [57] also employed language heuristics to craft user stories. This study introduces an approach that streamlines the semi-automated generation of user stories and Gherkin scenarios from process models, utilizing transformation patterns. In this method, they utilize the Business Process Modeling Notation (BPMN) to represent business process models, and the transformation patterns are constructed based on the partial metamodels of BPMN.

Li et al. [48] relied on graph definition using the Resource Description Framework (RDF) language and chatbot conversation to gather user stories. The paper proposes a conversation scheme able to transform the user's real life into a user story graph based on a series of first-order logic predicates. The usage of RDF language allows an easy transfer of learned knowledge into simulation environments by domain-specific controlling systems.

Veitia et al. [12] worked to identify user stories in the issue tracker corpus. They trained two different binary models of Neural Networks using Tensor Flow[10] and executed a comparison between both. The first model is a long short-term memory (LSTM) bidirectional neural network and the second one is a pre-trained language model called ElMo [65]. The results identified the model ELMo as being the best fit for the problem posed highlighting the potential of large language models (LLM) for this task [66]. It classified issues in user stories with an efficiency of approximately 96%.

Resketi et al. [49] made use of BoW (Bag of Words) to extract keywords and verbs from a set of user stories and write new ones. They also used tokenization, filtering, stopword removal, and verb parser. The paper has a complete evaluation procedure composed of three different experiments, the first one quantitative, and the other two qualitative. They reached 97% of micro F-measure and 93% of macro F-measure, which is promising and concluded by the expert's feedback that these new user stories can be used as the base user stories in future similar projects.

Panichella et al. [51] proposed the tool Requirements-Collector, a machine and deep learning-based tool for automating the tasks of requirements specification. This tool made use of a wide range of algorithms for this purpose, such as J48 (C4.5), PART, NaiveBayes, IBk (KNN), OneR, SMO, Logistic, AdaBoostM1, LogitBoost, DecisionStump, LinearRegression and RegressionByDiscretization. They used stakeholder meeting transcripts to automatically generate user stories and classify user reviews. The preliminary results highlight its accuracy for requirements extraction, though this is an ongoing work that still needs more evaluation. In the end, this is an exciting example of using different ML techniques to identify and specify system requirements.

Nistala et al. [52] proposed an approach for generating context-sensitive user stories from diverse specifications. This is a complete work committed to using a combination of NLP techniques to convert different formats of requirements specifications into user stories that can seamlessly be linked up to other tools. The process proposed is divided into five steps. The first one is pre-processing, able to remove typographical errors, style errors and taxonomic variation. So, they parse the documents using Open NLP Parser[11] and Docx4j.[12] Next, they make a post-processing step using a classification algorithm combined with regex and heuristics. Then, they populate a model able to generate user stories as output and create the knowledge query that will allow the user to consult the model.

Kumar et al. [53] proposed an approach to split complex user stories, so that implementation can be done quickly reducing the development time of software. They used machine learning clustering to evaluate the cohesion between user stories and decide whether they need to be split into smaller ones. The algorithms chosen for this experiment are K-means and K-medoids, being that K-means give a more remarkable output than K-medoids in this case. They performed the user story splitting manually based on CRUD and data entry breaking approaches. It is possible to run the clustering procedure as many times as needed to evaluate the cohesion of a set of user stories. If the cohesion between user stories is good then implementation becomes also easy for developers.

Table 8 presents a list of NLP and ML techniques used in the primary studies identified by our SLR. There is a notable presence of linguistic heuristics and regex algorithms to deal with user stories. They take advantage of the user story templates to break down the sentences or retrieve data. Although there is a strong presence of NLP approaches, it is also interesting to note that different ML algorithms were used by the authors, and chatbots were employed to support the user story elicitation process.

5.4 (RQ3) How user story quality is evaluated?

Different ways exist to evaluate the user story quality as described in Sect. 3.2. The goal of this research question is to understand which techniques were applied by the authors to measure the overall quality of user stories automatically

[10] https://www.tensorflow.org.

[11] https://opennlp.apache.org/.

[12] https://www.docx4java.org/trac/docx4j.

Table 8 Natural language processing and machine learning techniques used in primary studies to identify and generate user stories automatically

Technique	Category	Primary studies
Bag of words	NLP	[49]
Chatbot	Tool	[48, 54, 55]
Classification	SL	[10, 50, 51]
Clustering	UL	[53]
Docx4j	Tool	[52]
Graphs	NLP	[48, 57]
Language heuristics or regex	NLP	[11, 45, 47, 49, 52, 54, 55, 57]
Lexical dictionaries	NLP	[58]
NER	NLP	[11, 50, 58]
Neural Networks	UL	[12]
Ontology	NLP	[45, 46, 56]
POS tagging	NLP	[11, 58]
Text Parser	NLP	[49, 52, 58]
Tokenization	NLP	[11, 49, 55]
Sentiment analysis	SL	[50]

To better understand the table: *NLP* Natural Language Processing, *SL* Supervised Learning, *UL* Unsupervised Learning

generated. Although some of the approaches are well-defined and used in academia, we wanted to verify whether the authors relied on them as an evaluation step in their works. Overall, the adoption of RE quality guidelines to evaluate user stories generated is low, though some papers made use of quantitative methods to evaluate the proposed approaches. Table 9 summarizes our findings.

Numerically, seven of the 17 studies (41%) [45, 47, 48, 50, 51, 56, 57] did not present any type of quality evaluation. The major part of these studies are short papers only committed to presenting their approaches without validation. Five studies (29%) [10, 12, 53, 55, 58] are based on only quantitative measures for validation, without running any controlled experiment with humans or RE guidance. In addition, the other four papers (24%) [46, 49, 52, 54] used some qualitative approaches to check the characteristics of the user stories generated. Finally, one paper (6%) [11] made only an empirical evaluation of the dataset of user stories resulting based on the author's experience in RE. It is also worth mentioning that Heng et al. [56] performed a guided interview with software specialists to validate the ontology proposed for the generation of user stories and BDD scenarios.

Rodeghero et al. [10] used the standard machine learning performance metrics (*precision, recall, true-positive rate, false-positive rate, and pyramid precision*) to evaluate the quality of the extracted data compared with the manually annotated data. By using these metrics, the authors could find the best training configuration for user story data classification. Veitia et al. [12] followed a similar procedure to validate their machine-learning approach. In this case, the authors relied on neural networks to classify user stories.

Kumar et al. [53] also used an automated statistical evaluation for their results. They compared the *silhouette value* between non-split and split user story sets to measure the cohesion among user stories. The higher the *silhouette value* higher the cohesion. Dwitam et al. [55] used several metrics to evaluate the performance of the proposed chatbot, such as *total elapsed time, total number of user turn, and total number of system turn*, among others. They also surveyed to investigate how the users were prone to use the chatbot application proposed by them. However, they do not evaluate the quality of the user stories generated by the chatbot application.

Siahaan et al. [58] manually labeled semantic role based on the aspects of *who*, *what* and *why* in the news dataset. So, they performed the automatic extraction using the NLP techniques and based their evaluation on usual statistic measures of precision, recall and F-measure comparing the elements extracted with those manually annotated.

Murtazina et al. [46] used specific rules based on a domain ontology to validate the user stories created. These rules validated the presence of the main elements of a user story: role, action and benefit. The ontological approach differs from others because it can be easily applied to new and existing artifacts to ensure the quality of the requirements. It is valuable for agile environments where changes are welcome in different steps of the development process.

Resketi et al. [49] made use of the AQUSA tool (based on the QUS framework) as a preprocessing step to check the quality of the user stories used in their experiment and improve the user stories generated by their approach. The final set of user stories generated is evaluated statistically and

Table 9 Summary on how the quality of user story generated is evaluated by the primary studies

Study	Quality evaluation	RE framework or tool
Thamrongchote et al. [45]	–	–
Rodeghero et al. [10]	Quantitative	–
Murtazina et al. [46]	Qualitative	Ontology
Santos et al. [47]	–	–
Raharjana et al. [11]	Empirical	–
Li et al. [48]	–	–
Veitia et al. [12]	Quantitative	–
Resketi et al. [49]	Qualitative	QUS/Survey
Henriksson et al. [50]	–	–
Panichella et al. [51]	–	–
Nistala et al. [52]	Qualitative	–
Kumar et al. [53]	Quantitative	–
Lam et al. [54]	Qualitative	ISO 9126
Dwitam et al. [55]	Quantitative	–
Heng et al. [56]	–	–
Mateus et al. [57]	–	–
Siahaan et al. [58]	Quantitative	–

The symbol "–" means *none* or *unspecified*

by software practitioners through a survey. This paper brings a real concern about the user stories' quality considering their evaluation before and after the approach proposed.

Nistala et al. [52] relied on the expert's experience to validate the feature and user stories produced by the data model proposed. The experts gave detailed feedback (scores and comments) based on a comparison of the generated output with the original data from the source specification document. The main focus of this analysis was to check the correctness of the extracted sentences.

Lam et al. [54] utilized a controlled experiment to validate the user stories generated by their chatbot application. They used the survey approach based on ISO 9126 to measure completeness, correctness, and verifiability. The survey was designed using a 5-point Likert scale (0 = Totally disagree, 1 = Disagree, 2 = Neither agree nor disagree, 3 = Agree, 4 = Strongly agree), composed of five questions. They concluded that their system was able to create user stories more complete, correct and verifiable than using traditional RE techniques.

6 Discussion

Software requirements can emanate from various sources and exist in different formats, reflecting the diverse nature of the development process [9]. Stakeholders, such as clients, end-users, and domain experts, often play a crucial role in identifying functional and non-functional requirements. These requirements are gathered through interviews, surveys, workshops, and user feedback. While requirements manifest in various formats, user stories are one of the preferred methods of software developers because of their convenience and simplicity [22].

Due to time constraints, inexperience, and other reasons discussed in this paper, manually defined user stories often lack quality and consistency [7, 67]. High-quality user stories are mandatory to ensure effective communication and comprehension among the development team and stakeholders.

Through this SLR, we have confirmed the interest in research on automated writing of user stories. Albeit still in its infancy, this topic promises to improve software development in three significant ways. First, by employing efficient techniques and tools, teams can expedite the gathering of requirements from stakeholders, reducing the overall project duration. Saving time in RE and specification steps is essential for efficient and streamlined software development processes. Second, assuming that the generating method is accurate, it would undoubtedly contribute to enhancing the overall quality and ensuring a more consistent writing style across the project. Last but not least, it would result in more time spent on the development instead of on refinement meetings. Consequently, teams could get faster feedback from their clients and be more agile in the broad sense.

6.1 RQ1: Discussing the availability of user stories datasets

The small number of studies and research papers, however, gives a glimpse of the headwinds faced to make this a reality. First, we have seen that very few datasets are publicly

available to conduct such research. As we are shifting toward machine learning-based approaches, it is increasingly critical that researchers and practitioners share their datasets; both to further enhance repeatability and improve the quality of the research results.

Indeed, among the papers surveyed, most of them are short papers with limited validation or none at all. In some of them, the corpus is missing, and in others, there is no experiment done to validate the approach. Therefore, it seems clear that generating user stories automatically is an open topic at the crossroads of RE and AI.

In ML, the requirement for bigger datasets is closely correlated to the data and the model complexity. Text is well-known as one of the highest dimensionality data occurring in real-world problems [68]. Small datasets may also hinder model generalization and provoke overfitting [69]. Moreover, the lack of datasets slows progress, making it harder to evaluate ML models in different scenarios. In turn, superficial results prevent the adoption of those approaches by the RE community.

The resolution to the challenges we face may lie in adopting a transdisciplinary approach, one that intersects the realms of Artificial Intelligence (AI) and Requirements Engineering (RE) research. Currently, there is a palpable excitement surrounding AI, which has sparked motivating interest within software development companies. This enthusiasm underscores the potential of AI tools.

Software companies have data warehouses of different types of requirements artifacts. For instance, project tracking software stores vast quantities of user stories, requirements, business rules, and other pertinent data. Leveraging these rich datasets could serve to empower the training of various machine learning models, thereby offering invaluable assistance to RE practitioners.

RQ1 Findings

- The creation of new publicly available datasets to support projects focused on generating user stories appears to be a research opportunity in the area.
- The authors utilized datasets of varied formats and sources, which proved instrumental in generating user stories. This versatility can be attributed to the inherent nature of user stories, which often stem from diverse sources across different stages of software projects.
- Notably, some papers publicly share their datasets [10, 12, 49, 53, 56, 58].
- The identified datasets publically available are Meetings Transcripts, Software Issue Tracker CONNECT, Requirements dataset (user stories), PURE and US/BDD scenarios.

6.2 RQ2: Discussing the approaches explored by the primary studies

Regarding the approaches explored by the primary studies, it is possible to identify a variety of different techniques employed, many of them combined to achieve better results in automatically defining user stories. We classified the techniques used in four different groups: Natural Language Processing, Supervised Machine Learning, Unsupervised Machine Learning and Tools (Table 8).

6.2.1 Preprocessing methods

In terms of NLP, the papers surveyed explored a diverse set of techniques to identify and generate textual user stories. Absolutely all the papers that proposed some approach used natural language processing in at least one step of the process to automatically identify or generate user stories. NLP provides varied techniques that can be combined with language heuristics to reach different goals. It can also be used as a preprocessing step to machine learning approaches.

It is quite common to observe well-known NLP techniques such as Tokenization, POS tagging, Text parser, Named Entity Recognition (NER) and Bag of words being cross-applied for preprocessing text in different scenarios. Some papers relied on these techniques to break down the natural language and then propose different language heuristics to identify and produce user stories.

Tokenization is the act of breaking down the text into smaller units (e.g., words, letters, etc.). This technique is generally essential in NLP or ML and many libraries and frameworks offer this algorithm off-the-shelf, such as NLTK[13] and spaCy[14] [70].

Part-of-speech or POS tagging is a process that involves tagging each token of a sentence with a part-of-speech tag. This tag represents the grammatical class of the token, as a verb, noun, preposition, postpositions, adverbs, and so on. These tags are based on an international standard called *universal part-of-speech tagset*[15] [70].

The text parser technique is used to analyze the structure of a sentence, and it is a compliment to the POS tagging. There are two types of text parsers, the *constituency parsing* and the *dependency parsing*. The first one is used to break down the structure of a sentence according to its grammatical classification. For example, we can divide a sentence into two parts: a noun phrase (NP) followed by a verb phrase (VP), and so break each part into smaller grammatical portions. The second type of parser can specify the grammatical relation among the tokens of a sentence, using directional arrows to

[13] https://www.nltk.org/.

[14] https://spacy.io/.

[15] http://realworldnlpbook.com/ch1.html#universal-pos.

indicate the structural dependency [70]. The combination of the text parser with the POS tagging can be useful to identify parts of the user story by their syntax meaning. A verb, for example, can represent the action part of a user story. Nouns can represent the role or even some chunk of the benefit.

Named-entity recognition (NER) algorithms identify and classify named entities within text, such as people, organizations, locations, and titles. NER can be leveraged to enhance role identification in the user story composition.

The Bag-of-Words (BoW) technique is a method for representing text documents by analyzing the frequency of individual words, ignoring their order and, generally, their grammar. While it does not capture the meaning of sentences, BoW can be useful for identifying prominent themes and topics within a collection of text documents. Hence, they can be a starting point for writing valuable user stories using the highest-ranked tokens. It is also generally used in conjunction with a Term-Frequency-Inverse Document Frequency [71] to weigh the importance of words.

It is important to highlight that NLP techniques alone are not enough to extract and write user stories. Usually, it is necessary a combination of techniques to produce such results. The majority of the authors relied on their own algorithms supported by language heuristics and NLP techniques to identify, collect and write user stories. A cross-technique approach seems to be the best option for those using NLP to deal with user stories.

6.2.2 Knowledge representation

Using ontologies is a clever approach to organizing user story data. Informally, the ontology of a certain domain defines its terminology, concepts, classification, taxonomy, axioms and relations. In a nutshell, ontology is an important part of the knowledge about any domain [72]. However, they suffer from the well-established drawbacks of ontology-based AI; demand of domain experts, difficulties in setting universally accepted concepts and relationships, limited lexical coverage, and the difficulties in their upgrading [73].

As software can be seen as an abstraction of the real world, ontologies can play an important role in establishing the elements and properties of the domain to empower software engineers to better understand the context in which the software will be used. As user stories are split into three different parts (role, action, benefit) it makes it easy to organize data into categories and reuse data to create new and different user stories. Ontologies have long been employed to support software requirements elicitation, as can be read in the study of Kaiya *et. al* [74].

At the same level of knowledge representation, we found the use of semantic role labeling technique to identify the aspects of *who, what* and *why* in user stories. Semantic Role Labeling (SRL) is a natural language processing task that involves identifying the semantic roles of words or phrases in a sentence and assigning them to specific syntactic constituents. These roles represent the different functions that words play to the sentence's main verb, such as agent, patient, or instrument [75]. Raharjana et al. [11] and Sianhaan et al. [58] relied on their specific algorithms to identify semantic aspects of user stories.

The user story graph proposed by [48] is an example of how user stories are connected and dependent. One of the tasks performed by the authors used first-order logic predicates to identify the relationship among user stories. A better understanding of these relationships can assist software engineers in splitting and prioritizing user stories while working in agile environments.

6.2.3 Classic machine learning

Our machine learning analysis is divided into two segments: supervised and unsupervised learning. Supervised learning is the classic task of learning from examples with *ground truth* to recognize the errors of the model and revise it. Some studies [10, 50, 51] performed text classification using a supervised approach to identify user story information in different requirements datasets. Identifying user story information can be seen as an elicitation procedure able to save the efforts of the development team in this task. The referenced papers made use of different algorithms for this goal and also made different use of the classified data. The drawback of this approach can be the corpus dependency to train a reliable model able to correctly identify user stories in different projects. It might be necessary to include other algorithms in this equation to increase the accuracy of the model. For example, Henriksson et al. [50] also relied on NER to support the identification of roles, and sentiment analysis to identify the benefit of the user story.

The usage of sentiment analysis was cleverly proposed by Henriksson et al. [50] to identify the type of emotion or attitude expressed in a chunk of a user story, including the possible value: positive, negative, or neutral. Based on a corpus of system user reviews, it is possible to identify the type of request made by the user. For example, a negative sentiment can indicate a system bug or a disagreement with some feature. In short, this analysis can further support the specification of new user stories.

Alternatively, unsupervised machine learning involves algorithms that learn patterns and structures without receiving explicit *ground truth*. They analyze unlabeled data samples to uncover hidden relationships, clusters, or anomalies within the data itself. For instance, Kumar [53] used ML clustering (*K-means* and *K-medoids* algorithms) as a tool to validate their approach to user story splitting. It is one more case in which unsupervised learning strategies can be used for model or approach evaluation. In addition,

after reading this paper we concluded that the cohesion comparison between two or more sets of user stories can also help software practitioners in validating their consistency and boundary, satisfying quality attributes of ISO/IEC/IEEE 29148:2011 [35].

6.2.4 Neural networks

Neural networks are a family of ML models inspired by the structure and function of their biological counterparts. They became famous in the last decade due to their inherent ability to learn good representations of complex data [76]. The study of Veitia et al. [12] employed them to identify user stories in a software issue tracker dataset. This is the only paper that made use of neural networks for this purpose. As a complement, the authors compared the results produced by a long short-term memory (LSTM) neural network with the ElMo language model [65]. ElMo is one of the first large language models (LLMs) to emerge in the literature. ElMo is based on bidirectional LSTMs. Like other LLMs, it is trained using massive amounts of unlabeled text through a collection of pretraining objectives to enable the model to *teach itself* about the text [77]. In a nutshell, the goal of an LLM is to correctly predict the probability distribution of the next word, given a sequence of words. LLMs were mostly popularized and formalized, however, by two keystone papers: BERT [78] and GPT [79]. Both are trained in different ways and can also generate text differently.

A considerable amount of models are continuously introduced [80–83] coming to address text generation proposing diverse parameters. Many of these models allow the user to fine-tune using specific corpora and so teach the model to generate answers more context-sensitive. For example, we can see studies trying to take advantage of the language models to address problems in software engineering [13–15], and so proposing different ways to fine-tune the models to work in this specific context.

Most specifically aligned with the context of this SLR, we see Jain et al. [84] evaluating different LLMs to automatically summarize the requirements present in software contracts in a language comprehensible to business analysts. This study concludes that Pegasus LLM generates the most accurate summaries with the highest ROUGE score as compared to other models. They also relied on the experience of different analysts to evaluate the summaries generated. From the point of view of these authors, the language models are sufficiently powerful to disrupt the research in RE and propose state-of-the-art approaches to assist software practitioners in their daily routines.

6.2.5 Tools

We classified chatbots as a tool as different hands-on mechanisms can be used to train chatbot systems. A chatbot is a software program designed to mimic human conversation, typically with a user. While not all chatbots incorporate AI capabilities, modern ones generally leverage conversational AI methods like NLP and LLMs to comprehend user queries and provide automated responses [85]. Indeed, in the past couple of years new chatbots being developed usually rely mostly on new large language models. Different chatbot builders can be used to create chatbots for an application. Lam et al. [54], for example, used the IBM Watson Chatbot system empowered by a web implementation to gather user story information. Additionally, there are other builders available in the market that make easier the implementation of chatbots, such as Dialogflow, Qnamaker and Woebot.

The usage of a chatbot is an interesting approach to gathering requirements and then converting them into user stories (also seen in [86]). The authors relied on the user story structure to organize the conversation flow driving the user to give the correct elements that will help the system propose the user stories.

Docx4j is another tool used by the authors to deal with requirements descriptions. This tool is an open-source (Apache v2) Java library for creating, editing, and saving OpenXML files. Nistala et al. [52] used Docx4j to manipulate Microsoft Open XML files and then extract natural language requirements to create machine-processable models to generate user stories. Docx4j is an example of how the authors can employ different libraries to gather requirements in different formats. One of the big advantages of automatically generating user stories is the capacity of the system to understand the software requirements emanating from diverse sources.

RQ2 Findings

- Most studies rely on the user story structure to gather information and/or generate sentences [11, 45, 46, 49, 52, 53, 55, 58].
- Natural Language Preprocessing techniques, such as POS tagging and text parsing are common ground among the studies and are effective in supporting the user story data classification.
- Machine learning classification algorithms can be used to support the data-gathering process to composite user stories [10, 12, 51, 53].
- Knowledge representation, such as ontologies and graphs can be used to organize user stories data [45, 46, 48, 57].
- Chatbots are an effective option for gathering user story information [54, 55].
- Generation of user story statements is mostly a combination of language heuristics and natural language processing techniques.

6.3 RQ3: Discussing the employment of user story quality guidelines

With regard to the quality of the user stories generated, we concluded that the studies are rarely relying on qualitative standards for user stories. We understand that there is a high number of short papers, and it might be one of the reasons why they do not perform any qualitative evaluation steps. However, would be recommended to at least cite the framework or guideline intended to be used to assess the final quality of the statements written through the approach proposed.

It is important to remember that bad requirements descriptions do not guide the development of good software products. For this reason, the practice of evaluating the quality of user stories is very widespread in the market by software practitioners as shown in [18]. From our perspective, the adoption of standard guidelines for evaluating the quality of user stories is an essential procedure in papers proposing the automatic specification of user stories. Only through this would be possible to assess whether the approach proposed can generate unambiguous requirements statements.

We found that some studies [10, 12, 53, 55] adopted quantitative measurement as a way to assess the user stories generated. Although they are not ideal as a final evaluation step, they are an important part of this process. The authors can rely on different guidelines and technical approaches [35, 38, 39] to assess the user stories in a qualitative means. Possibly a controlled experiment would be necessary to reduce the bias of the evaluation.

We would like to highlight two papers that rely on established guidelines to evaluate the quality of software requirements. Lam et al. [54] benefit from a survey approach based on ISO 9126 to measure completeness, correctness and verifiability. Plus, Resketi et al. [49] made use of a tool created to evaluate user stories quality: AQUSA tool (based on the QUS framework). This is the only work that relies on a predefined framework for user stories.

An empirical evaluation made by authors cannot be enough to evaluate the statements generated, but it is a valuable complement to statistical assessment. Empirical evaluations rely on the software practitioner's experience in writing requirements statements. Many specialists benefit from their own experience to evaluate the quality of user stories [18]. Though, would it be enough to evaluate an automated approach to specify user stories? Further controlled experiments may be necessary to ensure the effectiveness of the approach. Another possibility could be the usage of ontologies to evaluate the overall quality of user stories, as made by Murtazina et al. [46].

Therefore, we conclude that there is broad space to employ better qualitative guidelines when generating user stories automatically. Statistical evaluation may not be the better way to assess the final quality of user stories due to the importance of these statements to the software development lifecycle, and the consequences of bad requirements to the final software product.

RQ3 Findings

- Lam *et al.* [54] benefit from a survey approach based on ISO 9126 to measure completeness, correctness and verifiability.
- Resketi *et al.* [49] made use of the AQUSA tool (based on the QUS framework) to evaluate user story quality.
- Ontologies can be used to validate the structure of user stories generated [46].
- Authors could use more qualitative measurement to ensure that the user stories generated can be used for software practitioners.

6.4 Future work

Other related papers concluded that there is a high potential for the applications of AI in RE [28, 87]. For example, Zhao et al. [28] states that the adoption of NLP on RE is an active and thriving research area as it has garnered a considerable number of publications and extensive attention from various researchers, while Cheligeer et al. [23] consider that ML can support requirements activities both theoretically and practically. At the same time, the study mapping performed by Amna et al. [21] on user story research identified the advantages of using user stories in requirement elicitation, which include improved team productivity, software quality and faster delivery.

The automated generation of user stories appears to be a promising area of research; however, the studies identified in this SLR indicate that it is still in its early stages of development, as pointed out by the broad presence of short papers in our dataset. Overall, we detected that most authors are committed to improving the approach presented in their studies as future works. It demonstrates the broad range of AI techniques available for user story generation. Only three studies do not present any future work plans (18%) [10, 45, 46].

The lack of reliable and openly accessible corpora for assessing methodologies across various scenarios exemplifies the vast potential yet to be explored in this field. Researchers can greatly enhance replicability by sharing their requirements corpora with the community. Of course, they can also work with the focus of creating new corpora and datasets in this area.

It is important to note that no standardized format for corpora is used in producing user stories (RQ1). Given that user stories can originate from a variety of sources, it is essen-

tial to employ approaches capable of collecting user story information from diverse channels and organizing it for subsequent text generation. Raharjana et al. [11] and Siahaan et al. [58], for example, were able to retrieve user story context from online news datasets, while Veitia et al. [12] and Kumar et al. [53] used the same dataset of user stories in two different approaches. Raharjana et al. [11] proposed an extension of their study by exercising the model on various online news sources and with different case events. Veitia et al. [12] also proposed refining the employed dataset by increasing the number of cases and retraining the model to have more accurate results. By doing that, they will prove the scalability, reliability and accuracy of the proposed models; we believe that researchers could reuse known corpora in RE, such as PURE[16] and PROMISE[17] or even work to create their corpora using data gathered from project management systems.

In terms of approaches employed (RQ2), we can see a variety of techniques being used to automatically generate user stories. It demonstrates how researchers use available AI techniques to deal with RE problems, from pure NLP approaches to deep learning strategies. LLM is being largely adopted in different scenarios (e.g., [88, 89]). We believe that new studies can take advantage of these models to complement their approaches or even develop new language models specific to user stories' context. We also found promising the usage of ontologies to organize user story data [45, 46, 56] and the usage of lexical dictionaries to support the identification of user story data in datasets [58].

Due to the general lack of industrial evaluation [28], researchers could work to implement their approaches in a way that software practitioners benefit greatly from daily. The utilization of portfolio management tools, such as Jira,[18], is common ground in the software engineering market. Many of these tools offer the possibility of implementing plugins able to facilitate daily operations. In this sense, software engineers could take a lot of advantages of plugins able to automatically write user stories for them, and so proportionate industrial feedback to studies in the intersection of AI and RE.

Finally, the quality of the user stories generated is a cornerstone of this topic (RQ3). Some researchers did not evaluate the quality of their outputs as rigorously as could be desired. The low adoption of strict guidelines to evaluate user story quality demonstrates that researchers need to investigate well-known frameworks, tools and guidelines for this purpose. The authors could extend the papers published using more strict procedures for quality evaluation.

For quality evaluation, we recommend the usage of two frameworks designed to evaluate the quality of user stories: QUS framework [38] is based on 13 language quality criteria and INVEST [39] brings well-defined guidelines to assess the quality of user stories; we also point to the investigation of the Requirement Smells study proposed by Femmer et al. [34], where are defined language criteria to the identification of bad software requirements; completely, authors could rely on the ISO standard ISO/IEC/IEEE 29,148 [35] what defines quality attributes to requirements statements.

6.5 Threats to validity

In this Section, we enumerate the main threats to the validity of our study. We present our threats analysis based on well-known literature reference [90, 91].

Internal Validity: It might be possible that relevant primary studies are missed during the database search process. To mitigate this threat, we performed a hybrid two-step search strategy as recommended by Mourão et al. [41] involving both keyword-based and snowballing searches (Sect. 4.1). Furthermore, we tested and refined our search string as suggested by Kitchenham et al. [40] in order to include only relevant terms. We opted to define a broader search string to obtain a more significant number of primary studies. The search and selection of studies as well as the data extraction process were performed by the corresponding author. As recommended by Wohlin *et al.* [90], interpretation problems or doubts were discussed with the other authors through consensus meetings.

External Validity: Regarding generalization of the results presented in our study, we followed well-established guidelines [40] to perform SLR and to obtain the most appropriate search strategy [41, 42] aiming to maximize generalization. In Sect. 4, we detail our SLR protocol and make available online the selected studies and data extracted from them enabling the extension of this SLR and consequently further mitigation of this threat.

Construct Validity: A possible construct validity could be related to the data manipulation activities such as the data extraction process and insufficient or incorrect search strategy. Aiming to mitigate this threat, we include in our SLR protocol different measures such as the development of a data extraction form to mitigate inconsistencies and extraction bias, the definition of IC and EC and the performance of quality assessment of the selected studies.

Conclusion Validity: This threat addresses the reasonability and credibility of the results and conclusions presented in this study. To mitigate this threat we systematically created the data extraction form emerging from the RQs (Table 5), we performed a broad thematic analysis and narrative synthesis [59, 60] to interpret and synthesize the extracted data to answer our RQs, and we conducted several meetings

[16] https://zenodo.org/records/7118517.

[17] https://zenodo.org/records/268542.

[18] https://www.atlassian.com/software/jira.

to discuss study findings and draw the correct conclusions. Further research can be performed to evolve the results and conclusion reported in this study.

7 Conclusion

This SLR provides an overview of the current research on the automatic generation of user stories for software development purposes. User stories are a prior approach to gathering software requirements in agile environments and can also be written relying on requirements artifacts. They are simple, easy to employ and able to embrace changes during the software development process, though they need to be well-written to communicate the right message to the development team.

Our SLR retrieved 17 papers addressing the automatic generation of user stories (as presented in Sect. 4). We evaluated all the papers as means to identify (i) which and how requirements corpora were used by the studies, (ii) which approaches were proposed and how they used ML and NLP to reach their goals, (iii) how these studies took care of the quality of the user stories generated.

We concluded that there is a scarcity of good corpora for experiments involving user stories. This makes it harder to develop new ideas and also difficult the replicability of the approaches proposed. Also, we identified a good broad of techniques being employed to automatically generate user stories, though the usage of well-known concepts of NLP and linguistic heuristics is presented in almost all studies. Finally, we close that the studies can use better guidelines to validate the quality of the user stories generated, though many of them propose quantitative metrics as evaluation.

In Sect. 6.4, we discussed some possible future work on this topic, with emphasis on the usage of language models and more efforts to evaluate the current approaches proposed for automatic generation of user stories. Despite what has already been accomplished, the best is yet to come.

References

1. Sommerville, I., Cliff, D., Calinescu, R., Keen, J., Kelly, T., Kwiatkowska, M.Z., McDermid, J.A., Paige, R.F.: Large-scale complex IT systems. Commun. ACM **55**(7), 71–77 (2012)
2. Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Manifesto for Agile Software Development (2001). http://www.agilemanifesto.org/ Accessed 30 April 2023
3. Ramesh, B., Cao, L., Baskerville, R.: Agile requirements engineering practices and challenges: an empirical study. Inf. Syst. J. **20**(5), 449–480 (2010)
4. Bourque, P., Fairley, R.E. (eds.): SWEBOK: Guide to the Software Engineering Body of Knowledge, Version 3.0 edn. IEEE Computer Society, Los Alamitos, CA (2014). http://www.swebok.org/ Accessed 30 April 2023
5. Cohn, M.: User Stories Applied: For Agile Software Development. Addison Wesley Longman Publishing Co., Inc, USA (2004)
6. Dalpiaz, F., van der Schalk, I., Brinkkemper, S., Aydemir, F.B., Lucassen, G.: Detecting terminological ambiguity in user stories: Tool and experimentation. Inf. Softw. Technol. **110**, 3–16 (2019)
7. Wagner, S., Fernández, D.M., Felderer, M., Vetrò, A., Kalinowski, M., Wieringa, R., Pfahl, D., Conte, T., Christiansson, M.-T., Greer, D., Lassenius, C., Männistö, T., Nayebi, M., Oivo, M., Penzenstadler, B., Prikladnicki, R., Ruhe, G., Schekelmann, A., Sen, S., Spínola, R., Tuzcu, A., Vara, J.L.D.L., Winkler, D.: Status quo in requirements engineering: A theory and a global family of surveys. ACM Trans. Softw. Eng. Methodol. **28**(2) (2019)
8. Fernández, D.M., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., Conte, T., Christiansson, M.-T., Greer, D., Lassenius, C., Männistö, T., Nayabi, M., Oivo, M., Penzenstadler, B., Pfahl, D., Prikladnicki, R., Ruhe, G., Schekelmann, A., Sen, S., Spinola, R., Tuzcu, A., Vara, J.L., Wieringa, R.: Naming the pain in requirements engineering. Empir. Softw. Eng. **22**(5), 2298–2338 (2017)
9. Aranda, G.N., Vizcaíno, A., Piattini, M.: A framework to improve communication during the requirements elicitation process in gsd projects. Requirem. Eng. **15**(4), 397–417 (2010)
10. Rodeghero, P., Jiang, S., Armaly, A., McMillan, C.: Detecting user story information in developer-client conversations to generate extractive summaries. In: Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017, pp. 49–59 (2017)
11. Raharjana, I.K., Siahaan, D., Fatichah, C.: User story extraction from online news for software requirements elicitation: A conceptual model. In: JCSSE 2019 - 16th International Joint Conference on Computer Science and Software Engineering: Knowledge Evolution Towards Singularity of Man-Machine Intelligence, pp. 342–347 (2019)
12. Peña Veitía, F.J., Roldán, L., Vegetti, M.: User stories identification in software's issues records using natural language processing. In: 2020 IEEE Congreso Bienal de Argentina, ARGENCON 2020 - 2020 IEEE Biennial Congress of Argentina, ARGENCON 2020 (2020)
13. Schröder, M.: Autoscrum: Automating project planning using large language models (2023) arXiV:2306.03197
14. Sridhara, G., G., R.H., Mazumdar, S.: Chatgpt: A study on its utility for ubiquitous software engineering tasks (2023) arXiv:2305.16837
15. Santos, C.A., Petrillo, F.: Towards auto-completion on software requirements statements (2021) arXiv:2106.13908
16. Goyal, R., Kumar, P., Singh, V.P.: A systematic survey on automated text generation tools and techniques: application, evaluation, and challenges. Multimedia Tools Appl. **82**(28), 43089–43144 (2023)
17. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change. Pearson Education, Boston, USA (2004)
18. Lucassen, G., Dalpiaz, F., Werf, J.M.E.M., Brinkkemper, S.: The Use and Effectiveness of User Stories in Practice. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **9619**, 205–222 (2016)
19. Cohn, M.: What is a user story template and why does it work so well? Mountain Goat Software (2019). http://www.agilemanifesto.org/ Accessed 25 April 2023
20. Pokharel, P., Vaidya, P.: A study of user story in practice. In: 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), pp. 1–5 (2020)
21. Amna, A.R., Poels, G.: Systematic literature mapping of user story research. IEEE Access **10**, 51723–51746 (2022)

22. Raharjana, I.K., Siahaan, D., Fatichah, C.: User stories and natural language processing: a systematic literature review. IEEE Access **9**, 53811–53826 (2021)
23. Cheligeer, C., Huang, J., Wu, G., Bhuiyan, N., Xu, Y., Zeng, Y.: Machine learning in requirements elicitation: a literature review. Artif. Intell. Eng. Design Anal. Manuf. (AIEDAM) **36**, 32 (2022)
24. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media Inc, Sebastopol, USA (2009)
25. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP Natural Language Processing Toolkit, vol. 2014-June, pp. 55–60 (2014)
26. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann, Elsevier, USA (2011)
27. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
28. Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K.J., Ajagbe, M.A., Chioasca, E.-V., Batista-Navarro, R.T.: Natural language processing for requirements engineering. ACM Comput. Surv. **54**(3), 55–15541 (2021)
29. Amna, A.R., Poels, G.: Ambiguity in user stories: a systematic literature review. Inf. Softw. Technol. **145**, 106824 (2022)
30. Weber-Jahnke, J.H., Onabajo, A.: Finding defects in natural language confidentiality requirements. In: 2009 17th IEEE International Requirements Engineering Conference, pp. 213–222 (2009)
31. Chantree, F., Nuseibeh, B., Roeck, A., Willis, A.: Identifying nocuous ambiguities in natural language requirements. In: 14th IEEE International Requirements Engineering Conference (RE'06), pp. 59–68 (2006)
32. Kurtanovic, Z., Maalej, W.: Automatically classifying functional and non-functional requirements using supervised machine learning. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 490–495 (2017)
33. Do, Q.A., Bhowmik, T., Bradshaw, G.L.: Capturing creative requirements via requirements reuse: a machine learning-based approach. J. Syst. Softw. **170**, 110730 (2020)
34. Femmer, H., Méndez Fernández, D., Wagner, S., Eder, S.: Rapid quality assurance with requirements smells. J. Syst. Softw. **123**, 190–213 (2017)
35. Standardization ISO, I.O.: ISO/IEC/IEEE international standard - systems and software engineering - life cycle processes - requirements engineering. ISO/IEC/IEEE 29148:2018(E) 1(1), 1–104 (2018)
36. Asadabadi, M.R., Saberi, M., Zwikael, O., Chang, E.: Ambiguous requirements: a semi-automated approach to identify and clarify ambiguity in large-scale projects. Comput. Ind. Eng. **149**, 106828 (2020)
37. Griva, A., Byrne, S., Dennehy, D., Conboy, K.: Software requirements quality: using analytics to challenge assumptions at intel. IEEE Softw. **39**(2), 80–88 (2022)
38. Lucassen, G., Dalpiaz, F., Werf, J.M.E.M., Brinkkemper, S.: Improving agile requirements: the quality user story framework and tool. Requirem. Eng. **21**(3), 383–403 (2016)
39. Wake, B., Wake, B.: Invest in good stories, and Smart Tasks (2003). https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/ Accessed 28 April 2023
40. Kitchenham, B.A., Budgen, D., Brereton, P.: Evidence-Based Software Engineering and Systematic Reviews. Innovations in Software Engineering and Software Development Series. Chapman & Hall/CRC, Boca Raton, USA (2015)
41. Mourão, E., Pimentel, J.F., Murta, L., Kalinowski, M., Mendes, E., Wohlin, C.: On the performance of hybrid search strategies for systematic literature reviews in software engineering. Inf. Softw. Technol. **123**, 106294 (2020)
42. Wohlin, C.: A snowballing procedure for systematic literature studies and a replication. In: International Conference on Evaluation and Assessment in Software Engineering (EASE), pp. 321–330 (2014)
43. Zhang, H., Muhammad, A.B.: Systematic reviews in software engineering: an empirical investigation. Inf. Softw. Technol. **55**, 1341–1354 (2012)
44. Wieringa, R., Maiden, N., Mead, N., Rolland, C.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirem. Eng. **11**(1), 102–107 (2006)
45. Thamrongchote, C., Vatanawood, W.: Business process ontology for defining user story. In: 15th IEEE/ACIS International Conference on Computer and Information Science. ICIS, pp. 1–4. IEEE Computer Society, Okayama, Japan (2016)
46. Murtazina, M., Avdeenko, T.V.: An ontology-based approach to the agile requirements engineering. In: Perspectives of System Informatics - 12th International Andrei P. Ershov Informatics Conference, PSI. Lecture Notes in Computer Science, vol. 11964, pp. 205–213. Springer, Novosibirsk, Russia (2019)
47. Santos, N., Pereira, J., Morais, F., Barros, J., Ferreira, N., Machado, R.J.: Deriving user stories for distributed scrum teams from iterative refinement of architectural models. In: Aguiar, A. (ed.) Proceedings of the 19th International Conference on Agile Software Development, pp. 40–1404. ACM, Porto, Portugal (2018)
48. Li, Y., Shibata, H., Takama, Y.: Chatbot-mediated personal daily context modeling upon user story graph. In: 2019 International Conference on Technologies and Applications of Artificial Intelligence. TAAI, pp. 1–6. IEEE, Kaohsiung, Taiwan (2019)
49. Resketi, M.R., Motameni, H., Nematzadeh, H., Akbari, E.: Automatic summarising of user stories in order to be reused in future similar projects. IET Softw. **14**(6), 711–723 (2020)
50. Henriksson, A., Zdravkovic, J.: A data-driven framework for automated requirements elicitation from heterogeneous digital sources. Lect. Not. Bus. Inf. Process. **400**, 351–365 (2020)
51. Panichella, S., Ruiz, M.: Requirements-collector: Automating requirements specification from elicitation sessions and user feedback. In: 28th IEEE International Requirements Engineering Conference. RE 2020, pp. 404–407. IEEE, Zurich, Switzerland (2020)
52. Nistala, P.V., Rajbhoj, A., Kulkarni, V., Soni, S., Nori, K.V., Reddy, R.: Towards digitalization of requirements: generating context-sensitive user stories from diverse specifications. Autom. Softw. Eng. **29**(1) (2022)
53. Kumar, B., Tiwari, U., Dobhal, D.C.: User story splitting in agile software development using machine learning approach. In: PDGC 2022–2022 7th International Conference on Parallel. Distributed and Grid Computing, pp. 167–171. IEEE, Waknaghat, India (2022)
54. Lam, L.K., Hurtado, C.A.L., Portillo, L.W.: Framework for automating requirement elicitation using a chatbot. In: Proceedings of the 2022 IEEE Engineering International Research Conference (EIRCON), pp. 1–4. IEEE, Peru (2022)
55. Dwitam, F., Rusli, A.: User stories collection via interactive chatbot to support requirements gathering. Telkomnika (Telecommun. Comput. Electron. Control) **18**(2), 890–898 (2020)
56. Heng, S., Tsilionis, K., Wautelet, Y.: Building user stories and behavior driven development scenarios with a strict set of concepts: Ontology, benefits and primary validation, pp. 1422–1429. ACM, Republic of Korea (2023)
57. Mateus, D., Silveira, D.S., Araújo, J.: A systematic approach to derive user stories and gherkin scenarios from BPMN models. Lecture Notes in Business Information Processing 483 LNBIP, 235–244 (2023)
58. Siahaan, D., Raharjana, I.K., Fatichah, C.: User story extraction from natural language for requirements elicitation: Identify

software-related information from online news. Information and Software Technology **158** (2023)
59. Cruzes, D.S., Dyba, T.: Recommended steps for thematic synthesis in software engineering. In: 2011 International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 275–284. IEEE Computer Society, Banff, Canada (2011)
60. Popay, J., Roberts, H., Sowden, A., Petticrew, M., Arai, L., Rodgers, M., Britten, N., Roen, K., Duffy, S.: Guidance on the conduct of narrative synthesis in systematic reviews. Technical report, A product from the ESRC Methods Programme (2006)
61. Ferrari, A., Dellorletta, F., Esuli, A., Gervasi, V., Gnesi, S.: Natural language requirements processing: a 4D vision. IEEE Softw. **34**(6), 28–35 (2017)
62. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. Language, Speech, and Communication. MIT Press, Cambridge, MA (1998)
63. Castillo-Barrera, F.E., Amador-García, M., Pérez-González, H.G., Martínez-Pérez, F.E., Torres-Reyes, F.J.: Adapting bloom's taxonomy for an agile classification of the complexity of the user stories in SCRUM. In: 2018 6th International Conference in Software Engineering Research and Innovation (CONISOFT), pp. 139–145. IEEE, San Luis Potosi, Mexico (2018)
64. Guarino, N., Oberle, D., Staab, S.: What Is an Ontology? International Handbooks on Information Systems, pp. 1–17. Springer, Switzerland (2009)
65. Peters, matthew e. and neumann, mark and iyyer, mohit and gardner, matt and clark, christopher and lee, kenton and zettlemoyer, luke (2018) arXiv:1802.05365
66. Gilardi, F., Alizadeh, M., Kubli, M.: ChatGPT outperforms crowd workers for text-annotation tasks. Proc. Natl. Acad. Sci. **120**(30) (2023)
67. Kustiawan, Y.A., Lim, T.Y.: User stories in requirements elicitation: A systematic literature review. In: 2023 IEEE 8th International Conference On Software Engineering and Computer Systems (ICSECS), pp. 211–216. IEEE, Penang, Malaysia (2023)
68. Görnitz, N., Youssef, A., Höppner, F.: What is the state of the art in word embeddings? In: Proceedings of the Workshop Events and Stories in the News (2015)
69. Géron, A.: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, pp. 1–856. O'Reilly Media, Inc., Sebastopol, USA (2019)
70. Hagiwara, M.: Real-World Natural Language Processing. Packt Publishing, USA (2020)
71. Jones, K.S.: A statistical interpretation of term specificity and its application in retrieval. J. Docum. **28**, 11–21 (1972)
72. Gaaevic, D., Djuric, D., Devedzic, V., Selic, B.: Model Driven Architecture and Ontology Development. Springer, Berlin, Heidelberg (2006)
73. Lastra-Díaz, J.J., Goikoetxea, J., Hadj Taieb, M.A., García-Serrano, A., Ben Aouicha, M., Agirre, E.: A reproducible survey on word embeddings and ontology-based methods for word similarity: linear combinations outperform the state of the art. Eng. Appl. Artif. Intell. **85**, 645–665 (2019)
74. 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 (2006)
75. Tan, Z., Wang, M., Xie, J., Chen, Y., Shi, X.: Deep semantic role labeling with self-attention. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), pp. 4929–4936. AAAI Press, New Orleans, Louisiana, USA (2018)
76. Pham, P., Nguyen, L.T.T., Pedrycz, W., Vo, B.: Deep learning, graph-based text representation and classification: a survey, perspectives and challenges. Artif. Intell. Rev. **56**(6) (2022)
77. Webber, E.: Pretrain Vision and Large Language Models in Python: End-To-end Techniques for Building and Deploying Foundation Models on AWS. Packt Publishing, Limited, United Kingdon (2023)
78. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019) arXiv:1810.04805
79. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training **1** (2018)
80. Fu, D.Y., Dao, T., Saab, K.K., Thomas, A.W., Rudra, A., Ré, C.: Hungry hungry hippos: Towards language modeling with state space models. In: The Eleventh International Conference on Learning Representations, ICLR. OpenReview.net, Kigali, Rwanda, (2023)
81. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A.M., Pillai, T.S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., Fiedel, N.: Palm: Scaling language modeling with pathways (2022) arXiv:2204.02311
82. Melis, G., Kočiský, T., Blunsom, P.: Mogrifier lstm (2020) arXiv:1909.01792
83. So, D.R., Mańke, W., Liu, H., Dai, Z., Shazeer, N., Le, Q.V.: Primer: Searching for efficient transformers for language modeling (2022) arXiv:2109.08668
84. Jain, C., Anish, P.R., Singh, A., Ghaisas, S.: A Transformer-based Approach for Abstractive Summarization of Requirements from Obligations in Software Engineering Contracts, vol. 2023-September, pp. 169–179 (2023)
85. What Is a Chatbot? IBM (2024). https://www.ibm.com/topics/chatbots Accessed 06 April 2023
86. Rajender Kumar Surana, C.S., Shriya, Gupta, D.B., Shankar, S.P.: Intelligent chatbot for requirements elicitation and classification. In: 2019 4th International Conference on Recent Trends on Electronics, Information, Communication and Technology (RTEICT), pp. 866–870 (2019)
87. Liu, K., Reddivari, S., Reddivari, K.: Artificial intelligence in software requirements engineering: State-of-the-art. In: 2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI), pp. 106–111 (2022)
88. Ronanki, K., Cabrero-Daniel, B., Berger, C.: Chatgpt as a tool for user story quality evaluation: Trustworthy out of the box? Lecture Notes in Business Information Processing 489 LNBIP, 173–181 (2024)
89. Scoggin, S.B., Torres Marques-Neto, H.: Identifying valid user stories using bert pre-trained natural language models. In: Rocha, A., Adeli, H., Dzemyda, G., Moreira, F., Colla, V. (eds.) Information Systems and Technologies, pp. 167–177. Springer, Cham (2024)
90. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer, Berlin, Heidelberg (2012)
91. Zhou, X., Jin, Y., Zhang, H., Li, S., Huang, X.: A map of threats to validity of systematic literature reviews in software engineering. In: 2016 23rd Asia-Pacific Software Engineering Conference (APSEC), pp. 153–160 (2016)