Received 27 April 2024, accepted 24 May 2024, date of publication 31 May 2024, date of current version 11 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3407753

RESEARCH ARTICLE

Natural Language Processing-Based Software Testing: A Systematic Literature Review

MOHAMED BOUKHLIF[1], MOHAMED HANINE[1], NASSIM KHARMOUM[2,3], ATENEA RUIGÓMEZ NORIEGA[4,5,6], DAVID GARCÍA OBESO[4,7,8], AND IMRAN ASHRAF[9], (Member, IEEE)

[1]LTI Laboratory, National School of Applied Sciences, Chouaib Doukkali University, El Jadida 24000, Morocco
[2]IPSS Team, Faculty of Sciences, Mohammed V University in Rabat, Rabat 10000, Morocco
[3]National Center for Scientific and Technical Research (CNRST), Rabat 10000, Morocco
[4]Universidad Europea del Atlántico, 39011 Santander, Spain
[5]Universidad Internacional Iberoamericana, Campeche 24560, Mexico
[6]Universidad Internacional Iberoamericana, Arecibo, PR 00613, USA
[7]Universidade Internacional do Cuanza, Cuito, Bié, Angola
[8]Universidad de La Romana, La Romana, Dominica
[9]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Republic of Korea

Corresponding authors: Imran Ashraf (imranashraf@ynu.ac.kr) and Mohamed Hanine (hanine.m@ucd.ac.ma)

This work was supported by European University of Atlantic.

ABSTRACT New approaches to software testing are required due to the rising complexity of today's software applications and the rapid growth of software engineering practices. Among these methods, one that has shown promise is the introduction of Natural Language Processing (NLP) tools to software testing practices. NLP has witnessed a rise in popularity within all IT fields, especially in software engineering, where its use has improved the way we extract information from textual data. The goal of this systematic literature review (SLR) is to provide an in-depth analysis of the present body of the literature on the expanding subject of NLP-based software testing. Through a repeatable process, that takes into account the quality of the research, we examined 24 papers extracted from Web of Science and Scopus databases to extract insights about the usage of NLP techniques in the field of software testing. Requirements analysis and test case generation popped up as the most hot topics in the field. We also explored NLP techniques, software testing types, machine/deep learning algorithms, and NLP tools and frameworks used in the studied body of literature. This study also stressed some recurrent open challenges that need further work in future research such as the generalization of the NLP algorithm across domains and languages and the ambiguity in the natural language requirements. Software testing professionals and researchers can get important insights from the findings of this SLR, which will help them comprehend the advantages and challenges of using NLP in software testing.

INDEX TERMS Software testing, natural language processing (NLP), systematic review, test case generation.

I. INTRODUCTION

In order to guarantee the dependability, functionality, and general quality of software systems, software testing is an essential phase in the software engineering process [1]. The importance of rigorous testing increases with the advancement of technology and the complexity of software programs. In addition to locating and fixing bugs and vulnerabilities, efficient software testing confirms that the program satisfies all criteria and operates without any issues in a variety of environments [2]. It acts as a vital defense against any malfunctions, security lapses, and user annoyance. Software testing helps create reliable, resilient, and user-friendly systems in addition to the immediate benefits of defect identification. This builds confidence in developers, stakeholders, and end users alike [3]. The value

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

of software testing in today's rapidly changing technological ecosystem goes far beyond simple error detection; it is essential to providing dependable, high-quality software solutions that satisfy a demanding clientele [4].

The use of artificial intelligence (AI) has revolutionized the world [9]. Its use in software testing signifies a revolutionary advancement in the search for effectiveness and accuracy [10]. The testing process is made faster and more thorough by the automation, adaptive learning, and predictive capabilities that AI algorithms can offer [11]. AI enhances software quality by optimizing testing procedures, minimizing manual labor, and generating intelligent test cases and automated anomaly detection [12]. AI and software testing work hand in hand to speed up the testing process and provide developers with the confidence and agility to handle the complex requirements of modern software engineering [13], [14].

A direction change has occurred with the use of Natural Language Processing (NLP) in software testing, which uses linguistic comprehension to improve many aspects of the testing environment [15]. Natural language requirements can be interpreted using NLP approaches, allowing for a more sophisticated and context-aware testing strategy. NLP adds a semantic understanding layer to testing, enabling intelligent test case design based on textual specifications and the relevant insights extracted from user response [16]. This combination of software testing with language understanding speeds up the process of finding ambiguities and inconsistencies and improves communication between development and testing teams [17]. NLP emerges as an effective companion as the software industry travels the complexities of many linguistic areas, transforming software testing, validation, and refinement [18].

NLP-based software testing confronts a multitude of challenges and open issues that necessitate careful consideration and innovative solutions. Firstly, the inherent ambiguity and variability of natural language pose significant hurdles in accurately interpreting and analyzing textual artifacts, such as requirements documents, user stories, and software specifications. Additionally, the domain-specific nature of software engineering introduces further complexities, requiring NLP models to adapt and specialize to diverse linguistic patterns and terminologies [19]. Furthermore, the scalability and generalizability of NLP-based testing frameworks remain elusive goals, with existing approaches often struggling to cope with the complexities of large-scale software systems and diverse testing scenarios. Moreover, the interpretability and trustworthiness of NLP models in the context of software testing raise profound concerns regarding reliability, robustness, and ethical considerations [20].

A systematic literature review synthesizes current knowledge in an organized manner, identifies patterns, and highlights gaps in the literature, acting as a precise compass across the enormous expanse of study [21]. It provides a thorough overview of the current state of a certain subject by using rigorous procedures for study selection and analysis. This helps academics, practitioners, and decision-makers make intelligent choices on future directions and insights [22].

In this study, we followed a thorough and repeatable process in order to review 24 papers extracted from Scopus and Web of Science, these papers were published between 2013 and 2023. We extract all helpful insight about the state of NLP usage in software testing, from famous NLP techniques, to open challenges, going through machine and deep learning techniques used in the field.

The subsequent sections of this article are organized as follows. Section II will commence by discussing related works in the studied field, followed by an exposition of the research methodology used in this review in Section III, Section IV will present the obtained results. The ensuing Section V will give a comprehensive discussion of the findings, while Section VI will expound upon the limitations and future work of this study, and, finally, Section VII will encapsulate the conclusions drawn from this study.

II. RELATED WORK

In recent years, NLP has emerged as a promising approach to enhance various aspects of software testing, from test case generation to test result analysis. This systematic literature review aims to synthesize the existing body of knowledge in this domain. This review is not the first in the field, Table 1 presents a comparison between related work.

In [5], the findings exhibit accuracy scores spanning from 70% to 90%, implying a degree of reliability. However, this broad spectrum leaves room for questioning the consistency and robustness of NLP algorithms in the domain. Furthermore, the authors classified the studies based on their contributions which underscores the multifaceted nature of NLP's application in software testing. Yet, this diversification may evoke concerns about the absence of a unified focus, potentially scattering research efforts. The authors also categorized the studies based on research methods which offers structure and rigor, yet the resulting analysis opens the door to criticism regarding the quality and depth of research within the domain. They also listed the number of requirements artifacts processed and test cases generated providing quantitative insights. They explored the methods employed for evaluating NLP approaches stressing the importance of robust evaluation criteria. The main limitation of their study is excluding information about the domain of the application, and not mentioning the recurrent challenges.

In study [6], the authors included a detailed listing of NLP steps such as tokenization, parts-of-speech (POS) tagging, chunking, and parsing, which are routinely employed in the context of test case generation, this enumeration offers valuable guidance on the NLP techniques commonly adopted by researchers. Concurrently, they also cataloged a range of NLP tools and algorithms applied or proposed by scholars for the explicit purpose of generating test cases from initial requirements. However, a notable limitation is that the scope of this review is confined to publications within the ACM, IEEE, and Springer databases.

TABLE 1. Comparative table of related work.

Paper	Year	Type	# of papers	Timespan	Sources
NLP-assisted software testing: A systematic mapping of the literature [5]	2020	Journal	67	2001-2017	Scopus, Google Scholar
A Comprehensive Investigation of Natural Language Processing Techniques and Tools to Generate Automated Test Cases [6]	2017	Conference	16	2005-2016	Springer, ACM, IEEE
An overview on test generation from functional requirements [7]	2011	Journal	22	1988-2009	Google, Google Scholar, Scopus, EI Compendex, ISI Web of Knowledge, IEEEXplore, ACM Digital Library, and CiteSeerX
The Applications of Natural Language Processing (NLP) for Software Requirement Engineering - A Systematic Literature Review [8]	2017	Conference	27	2002-2016	IEEE, ACM, Springer, and Elsevier
This study	2023	Journal	24	2013-2023	Scopus, Web of Science

The study [7] focused on techniques, tools, and the quantification of test cases and furnished significant insights. Its emphasis on well-documented approaches is underscored, recognizing that the practical application of any approach is contingent upon its clear and comprehensive documentation, this attention to documentation is pivotal for promoting the reproducibility and dissemination of research within the software testing field. It also highlights information regarding the availability, license, and operating environment of the tools and techniques under examination. However, certain inherent limitations must be addressed. Primarily, the systematic review lacks explicitly defined inclusion and exclusion criteria. This absence of well-defined criteria may introduce ambiguity and subjectivity into the selection process, potentially affecting the comprehensiveness and objectivity of the review.

As for [8], the authors did a listing of software requirements areas where NLP techniques are frequently practiced, encompassing tasks such as requirement analysis, requirement prioritization, ambiguity removal, and more. This enumeration provides a valuable roadmap for understanding the domains within software requirements where NLP techniques have gained prominence. They also mentioned the NLP tools and algorithms commonly utilized by researchers in this domain, which will serve as a valuable resource for comprehending the diverse computational mechanisms and linguistic processing tools employed within the realm of software requirements analysis. However, the review did not explicitly mention the accuracy or efficacy of the NLP tools and algorithms discussed. This absence of accuracy assessment may leave one with questions regarding the reliability and robustness of the NLP techniques employed in software requirements analysis.

Despite the growing interest in leveraging NLP techniques for software testing, a clear understanding of the challenges and opportunities in this domain remains elusive. Existing literature primarily showcases the potential benefits of NLP-based testing methodologies but often overlooks the nuanced complexities and unresolved issues inherent in their practical implementation. Additionally, this field is growing at an exponential rate, requiring literature reviews every year [5]. Consequently, there exists a significant research gap that necessitates comprehensive exploration and analysis. This research seeks to address this gap by conducting a systematic literature review (SLR) focused on elucidating the challenges and open issues in NLP-based software testing. In this study, we aim to overcome some of the limitations of existing reviews in the studied field and provide more insights about the use of NLP techniques in software testing.

III. RESEARCH METHODOLOGY

A. REVIEW QUESTIONS

In order to systematically explore and synthesize the existing body of literature, it is imperative to establish a clear framework guided by well-defined review questions. The formulation of precise review questions is the foundational step in conducting an SLR as it shapes the direction, scope, and objectives of the review process. We want to affirm that throughout this study, we meticulously followed established SLR protocols to ensure methodological rigor and comprehensiveness. From the initial stages of defining inclusion and exclusion criteria to the systematic search and selection of relevant papers, we adhered to a structured and replicable approach. Furthermore, in constructing the search query, we employed the Population, Intervention, Comparison, and Outcome (PICO) framework, a well-established approach for formulating research questions and search strategies in systematic reviews that aims to ensure the comprehensiveness and relevance of the retrieved literature. Additionally, we employed a rigorous screening process to select papers that met the predefined inclusion criteria, including relevance to NLP-based software testing, full implementation details, and empirical evaluations. This systematic approach enabled us to identify and analyze a focused yet representative set of papers that contributed significantly to our understanding of the topic.

In this section, we articulate the review questions that guide the SLR, outlining the specific inquiries we aim to address. These questions not only serve as the compass for the review but also provide the reader with a comprehensive understanding of the key themes and objectives that underpin this study.

- **RQ1: What are the most popular NLP techniques used in the field?** What are the prevalent linguistic and semantic analysis methods and approaches that have gained prominence in the field? providing valuable insights into the techniques most frequently employed by researchers and practitioners in this domain.
- **RQ2: How explicit are the NLP algorithms presented?** Is there an in-depth analysis and description of the NLP algorithm? Which will make reusing the tool/approach and reproducing the results easy for researchers?
- **RQ3: What language does the NLP models support?** This question aims to clarify the linguistic scope of the NLP model, which impacts the generalization of the approach/tool.
- **RQ4: What are the trending testing contexts/types?** We aim to uncover the specific contexts and types of testing that are currently gaining prominence or evolving as significant areas of research and practice in the field. This will help researchers discover which software testing areas are most suitable for the use of NLP.
- **RQ5: What are the trending Machine (ML) and Deep Learning (DL) algorithms used alongside NLP?** This question aims to explore the cutting-edge ML/DL algorithms employed in the field. This will help researchers and practitioners stay abreast of ML/DL that might work best alongside NLP.
- **RQ6: What are the Tools and Frameworks used in this field ?** What software tools and frameworks are currently employed within the domain?
- **RQ7: How reliable are the methods/approaches?** What are the evaluation results such as precision, recall, and f-measure (when present) of the approaches and tools presented in the studied papers?
- **RQ8: What are the most used datasets?** Datasets play a pivotal role in training and evaluating models, influencing the robustness and generalizability of NLP applications in the software testing arena. The question at hand centers on discerning the most frequently employed datasets in the field.
- **RQ9: What are the recurrent open challenges?** This question aims to identify and examine the persistent and unresolved issues or obstacles within the domain. This will shed light on critical areas where further research and innovation are needed.
- **RQ10: What are the domains of the SUT?** The domains of the Software Under Test (SUT) are important to researchers to identify the domains that need more attention and those with domain-specific challenges. Which enables the development of tailored solutions to a specific industry need.

Each review question in this SRL forms a crucial part of a cohesive narrative that systematically explores various aspects of NLP-based software testing. The investigation into the most popular NLP techniques naturally leads to an examination of the explicitness of the algorithms presented, as understanding the prevalence of specific techniques shapes the level of detail provided regarding their implementation. Similarly, the inquiry into NLP model language support is closely intertwined with the examination of trending testing contexts and types, as the language support of NLP models influences their applicability in different testing scenarios. Furthermore, the exploration of trending machine and deep learning algorithms used alongside NLP techniques is inherently linked to understanding the evolving landscape of NLP-based software testing, as these algorithms enhance the capabilities and performance of NLP-driven testing methodologies. The assessment of tools and frameworks usage is closely tied to evaluating the reliability of methods and approaches employed in the literature, as the availability and adoption of robust tools and frameworks contribute to the reliability and reproducibility of NLP-driven testing techniques. Identifying the most used datasets provides insights into the empirical foundations of NLP-based software testing research while understanding recurrent open challenges allows us to contextualize the limitations and gaps in the existing literature, guiding future research directions and priorities. Finally, exploring the domains of the system under test provides valuable context for understanding the applicability and generalizability of NLP-based testing methodologies across different domains and industries.

B. INCLUSION/EXCLUSION CRITERIA

The formulation of precise inclusion and exclusion criteria is a pivotal aspect of conducting an SLR. These criteria provide a structured framework that delineates the boundaries of our review, guiding the systematic selection of relevant studies while excluding those that do not meet the predefined criteria. In this section, we expound upon the specific parameters and rationale that underlie our inclusion/exclusion criteria. Establishing transparent and well-defined criteria ensures the methodological rigor of our review and enhances its reproducibility. These criteria serve as a crucial lens through which we sift through the vast body of literature to identify studies that align with the objectives of our SLR. Through this process, we aim to maintain a high standard of quality and relevance in the selection of primary research sources, ensuring that our findings and conclusions are grounded in a systematically derived and representative body of evidence.

1) DATABASE SELECTION

We chose Web of Science (WoS) and Scopus as databases for conducting the SLR because they offer several compelling advantages. These databases are celebrated for their extensive coverage of academic literature, encompassing a diverse range of sources, including journals and conference proceedings. This breadth of coverage makes them well-suited for multidisciplinary research, ensuring access to a wide array of literature, which is particularly valuable for comprehensive SLRs. Furthermore, WoS and Scopus span a multitude of academic disciplines, making them an ideal choice for research inquiries that require cross-disciplinary perspectives, enhancing the diversity of sources available.

Equally crucial is the rigorous quality control applied to the content in these databases, ensuring high-quality, peer-reviewed material. This quality assurance underpins the reliability of studies selected for inclusion in our SLR, fostering a high level of academic rigor. WoS and Scopus provide advanced search capabilities, including Boolean operators and filters, enabling the creation of precise search queries, and enhancing the efficiency of study identification.

2) SEARCH QUERY

To build the search query we chose the PICO criteria [22]. To find out the keywords and strings from the review questions, the term PICO means Population, Intervention, Comparison, and Outcomes, when in our case:

- **Population**: NLP usage in software testing
- **Interventions**: NLP-based software testing techniques/methods/tools/frameworks.
- **Comparison**: Comparison of the studies
- **Outcomes**: Develop/Improve the usage of NLP in software testing

Thus we opted for this search query: **(“natural language processing” OR “NLP”) AND (“software testing” OR “software quality assurance” OR “software verification” OR “software validation” OR “software quality control”)**

We employed Boolean operators to create a comprehensive search strategy to identify studies related to the intersection of natural language processing (NLP) and software testing. Here’s a breakdown of the query:

- **(“natural language processing” OR “NLP”)**: This part of the query is enclosed in parentheses and uses the OR operator. It searches for papers that contain either “natural language processing” or its abbreviation “NLP” in their content. This helps to capture studies that reference NLP using different terminologies.
- **AND**: The AND operator is used to combine the first part of the query with the second part to ensure that papers must include both elements.
- **(“software testing” OR “software quality assurance” OR “software verification” OR “software validation” OR “software quality control”)**: This portion of the query uses the OR operator to search for papers that include any of the specified terms related to software testing. This ensures that the search is inclusive and captures studies that might use different terminology to describe software testing processes.

Combining these elements in the query aims to retrieve a comprehensive set of papers that discuss the intersection of NLP and software testing, including their applications, methodologies, challenges, and advancements.

3) INCLUSION/EXCLUSION STEPS

After defining the query and databases, we ran the query in both databases and got a total dataset of 453 papers. The next phase is removing the duplicate papers which were 73 papers, and leaving us with 380 papers.

Next, we applied the following criteria:

- Exclude papers that are not written in English (6 papers excluded)
- Exclude papers outside this timespan: 2013-2023 inclusive (37 papers excluded)
- Exclude studies that are not consistent journal papers (277 papers excluded).

Figure 1 shows a flow chart summarizing the inclusion/exclusion criteria steps.

C. QUALITY ASSESSMENT

In the pursuit of methodological rigor and the assurance of the reliability of this SLR, a critical step lies in the comprehensive evaluation of the quality of the selected studies. This section serves as the linchpin of this review, where each included study undergoes meticulous scrutiny to determine its methodological soundness and the trustworthiness of its findings. The objective of this section is to provide readers with a transparent and objective evaluation of the studies included in the review, ensuring that the conclusions drawn are based on robust and well-conducted research. Through a systematic and standardized assessment process, we endeavor to identify the strengths and limitations of each study, allowing us to weigh their contributions appropriately within the context of the research objectives. This rigorous quality assessment not only enhances the credibility of this SLR but also aids readers and researchers in discerning the reliability of the synthesized evidence, thereby reinforcing the validity and impact of the findings.

The quality assessment of the papers is presented in the manual review section of Figure 1. In that section, we were trying to answer three questions to assess the quality of the primary studies:

- Does the paper actually use NLP techniques in order to improve software testing?
- Is the process of presenting the tool/approach/method clearly defined (problematic identification, NLP techniques, solution architecture)?
- Is there any empirical evidence for the solution? Any case studies, empirical results, or evaluation results?

After applying the inclusion/exclusion criteria and doing the quality assessment of the papers. A total of 24 papers were obtained, which will be used in the rest of this SLR.

While the review may appear focused in comparison to broader surveys, it is imperative to note that the emphasis was on meticulously analyzing a select set of high-quality, full-implementation papers. We recognize the importance of ensuring that the included literature provides substantive insights and robust methodologies relevant to the domain of NLP-based software testing. As such, we applied stringent criteria to assess the quality and relevance of each paper, prioritizing those that offered comprehensive insights, empirical evaluations, and practical applications of NLP techniques in software testing contexts. By adhering to these criteria,

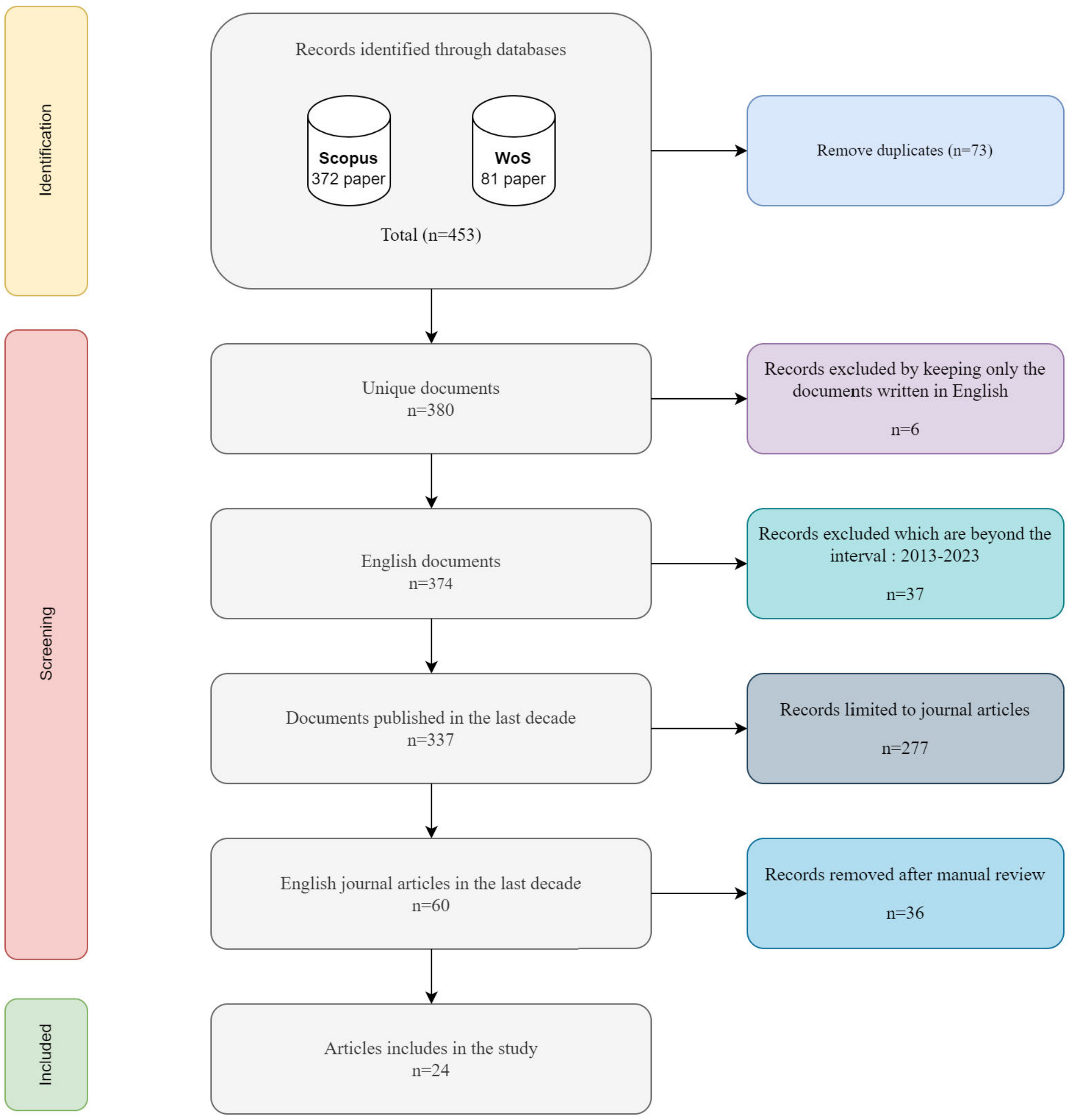

Identification
Screening
Included
Records identified through databases
Scopus
372 paper
WoS
81 paper
Total (n=453)
Remove duplicates (n=73)
Unique documents
n=380
Records excluded by keeping only the documents written in English
n=6
English documents
n=374
Records excluded which are beyond the interval : 2013-2023
n=37
Documents published in the last decade
n=337
Records limited to journal articles
n=277
English journal articles in the last decade
n=60
Records removed after manual review
n=36
Articles includes in the study
n=24

FIGURE 1. Flow chart of the inclusion/exclusion criteria.

we aimed to uphold the integrity and credibility of our review, thereby providing a solid foundation for synthesizing key findings and identifying recurring themes and challenges within the field.

D. PAPERS ANALYSIS

After applying the inclusion/exclusion criteria and quality assessment, we carried out an in-depth analysis of the 24 papers to extract any information that will help us answer the review questions. This information was classified in a spreadsheet with the columns in Table 2.

TABLE 2. The table of the extracted information.

Information	Review Question(s)
Software Testing Context	RQ4
Software Testing Type	RQ4
NLP Techniques	RQ1
NLP Model Language	RQ3
Machine/Deep Learning Algorithms	RQ5
Application Domains	RQ10
Dataset	RQ2, RQ8
Limitations	RQ9
Open Challenges	RQ9
Experimental Results	RQ7
Source Code Availability	RQ2
Software Tools and Frameworks	RQ6

E. FINAL PAPERS DATASET

As shown in Figure 2, over the last decade, NLP-based software testing has witnessed a notable surge in research activities, particularly in the latter years. The early years, spanning from 2013 to 2016, showed minimal research output with no published papers or only a marginal increase in

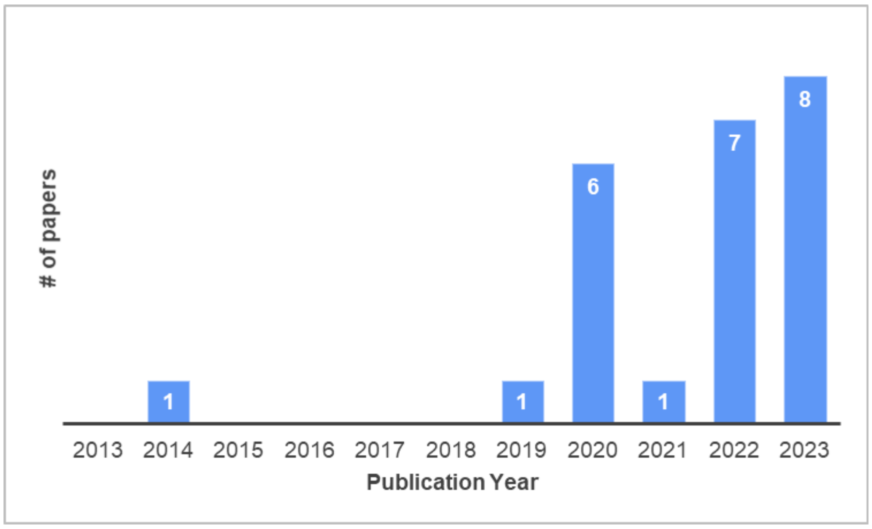


FIGURE 2. Field growth over the last decade.

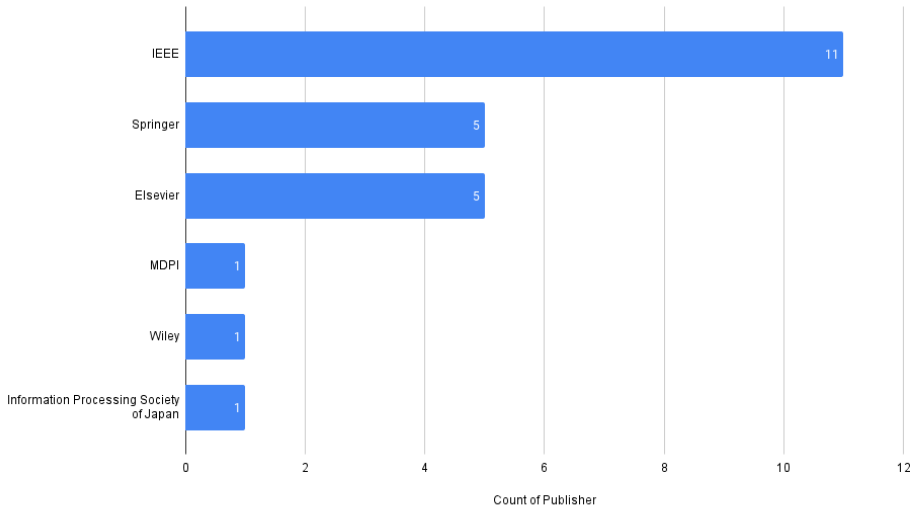


FIGURE 3. Publishers.

2014. However, starting in 2019, there was a discernible uptick, marked by an increasing trend in the number of papers. In 2019, there was a modest resurgence with one publication, and this growth gained momentum in 2020, with a noteworthy six papers. The year 2021 maintained momentum with one additional paper, and the trend reached its pinnacle in 2022 and 2023, demonstrating a substantial leap with seven and eight papers, respectively.

As shown in Figure 3, the studied papers were published by a variety of publishers. IEEE emerges as the predominant publisher, with 11 papers contributing to its repository. Following closely are Springer and Elsevier, each hosting 5 papers, indicating a balanced representation in academic platforms. Wiley accounts for 1 paper, as does MDPI, highlighting the diversity of sources. Additionally, the Information Processing Society of Japan is associated with 1 paper. This distribution across multiple publishers underscores the collaborative and expansive nature of academic exploration in the field, drawing from a range of reputable sources to enrich the body of knowledge in the field.

IV. RESULTS

A. RQ1: WHAT ARE THE MOST POPULAR NLP TECHNIQUES USED IN THE FIELD ?

In recent research papers, a diverse array of NLP techniques has been employed to address a variety of linguistic challenges. Tokenization, a fundamental technique for breaking down text into individual units, was frequently utilized, emphasizing its importance in many NLP tasks [23]. POS tagging, stop word removal, and term frequency-inverse

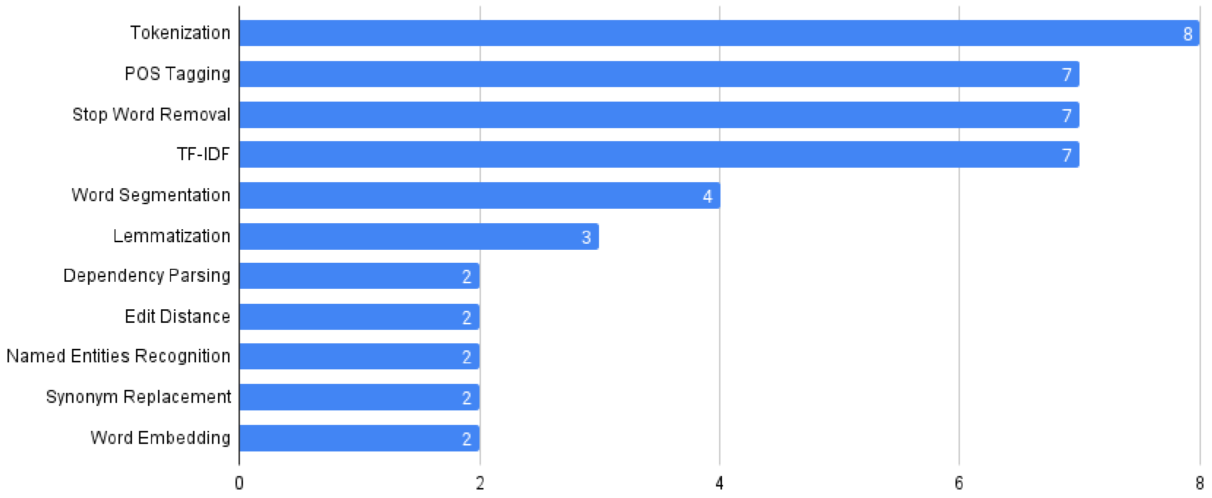


FIGURE 4. Most used NLP techniques.

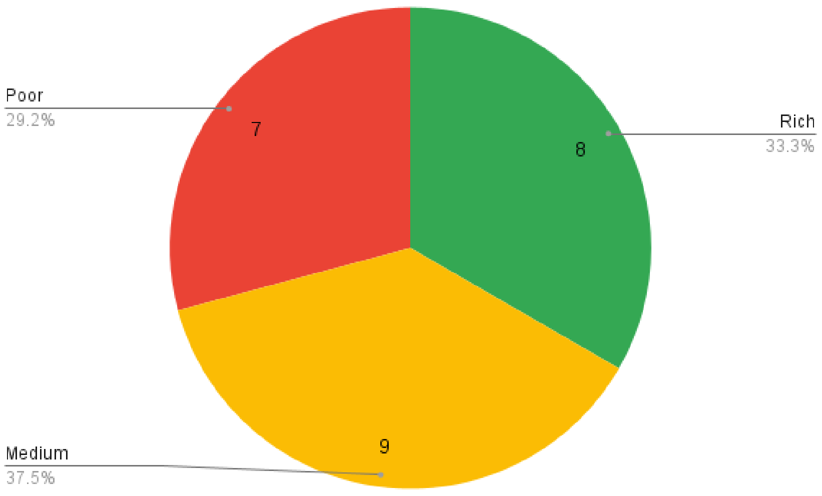


FIGURE 5. Explicitness of the NLP algorithms.

document frequency (TF-IDF), each employed in seven papers, showcase their pivotal roles in enhancing language understanding and information retrieval processes [24]. Word segmentation, explored in four papers, reflects the significance of this technique, particularly in languages with non-delimited words [25]. Lemmatization and dependency parsing, examined in three and two papers respectively, contribute to syntactic and morphological analysis, providing a deeper understanding of linguistic structures [26]. Edit distance and named entities recognition, each explored in two papers, demonstrate their relevance in applications requiring similarity measurement and entity identification, respectively [27]. Synonym replacement and word embedding, also examined in two papers each, underscore the growing interest in semantic enrichment and vector space representations [28], reflecting the dynamic landscape of NLP research. Figure 4 summarizes the NLP techniques mentioned.

B. RQ2: HOW EXPLICIT ARE THE NLP ALGORITHMS PRESENTED?

In this section, we assess the clarity of the NLP algorithms presented in the papers. We rank them in three categories: 1) Rich, for papers having an in-depth explanation and presentation of their NLP algorithm; 2) Medium, for papers with some details, but not a very deep explanation; 3) Poor, for papers with almost no details about the implementation of their NLP algorithm. In this ranking, we primarily consider the availability of the source code of the solution.

Figure 5 represents a pie chart depicting the explicitness of NLP algorithms in the studied research papers and offers valuable insights not only about transparency and openness but also about the possibility of reuse of the tools and methodologies. It is noteworthy that a significant portion, 9 papers (37.5%), falls under the ''Medium'' category, indicating papers that provide a level of detail

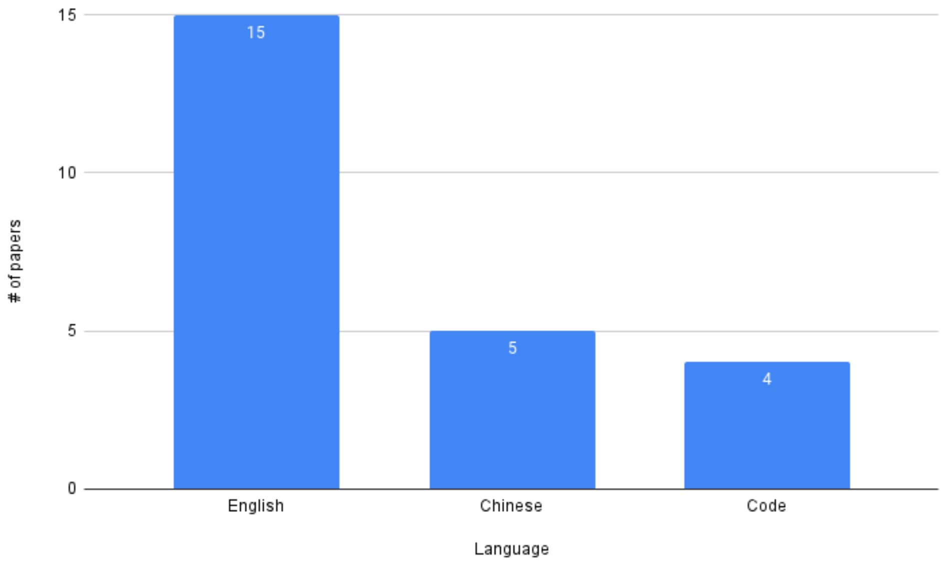


FIGURE 6. NLP models language support.

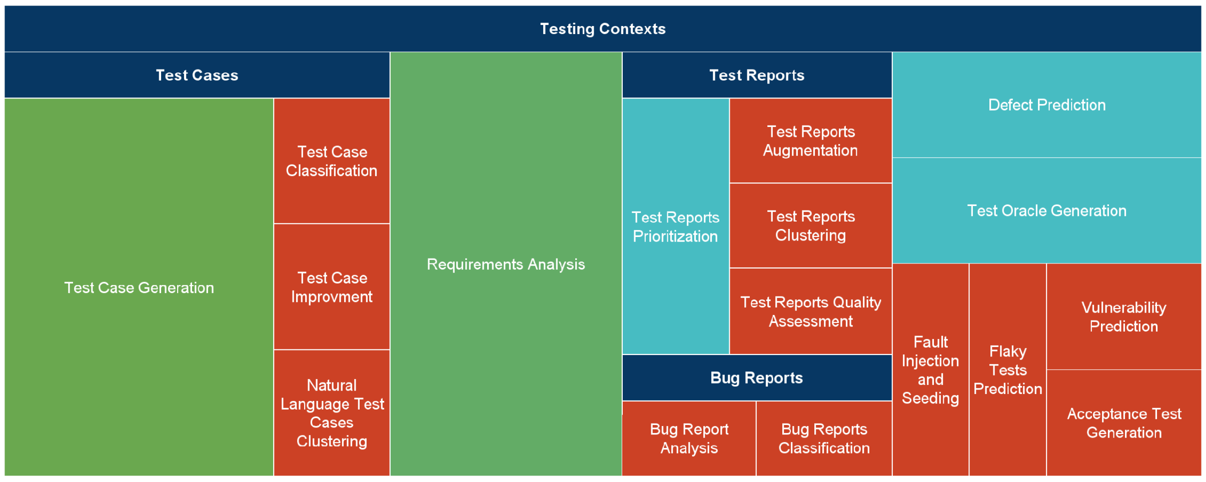


FIGURE 7. Treemap of software testing contexts.

about the implementation without reaching the level of full openness. The "Rich" category, comprising 8 papers (33.3%), signifies a commendable proportion of papers that openly share tools and methods, showcasing a commitment to transparency and reproducibility, most of the papers of this category open source their solution. On the other hand, the "Poor" category, representing 7 papers (29.2%), points to a non-negligible portion of papers that offer limited details about their implementations. This distribution underscores the importance of encouraging comprehensive documentation and openness in research, fostering a collaborative and reproducible environment within the NLP community. 10 papers are sharing the source code related to their method, approach, tool: [29], [30], [31], [32], [33], [34], [35], [36], [37], [38].

C. RQ3: WHAT LANGUAGE DOES THE NLP MODELS SUPPORT?

Language support is a fundamental aspect of NLP research that significantly influences the generalizability and real-world impact of models [39]. In this section, we map the language support in the studied papers. Regardless of the NLP tool used, those languages are extracted from the validation phase of the studies (case study, application example…).

The language support distribution in the studied research papers is outlined in Figure 6. Among the papers, 15 are focused on the English language, indicating a substantial emphasis on English in the literature. Additionally, Chinese, as a language of focus, is represented in 5 papers, reflecting a notable but comparatively smaller proportion. Furthermore, an intriguing aspect is the inclusion of 4 papers that

TABLE 3. Software testing contexts.

Software Testing Context	# of papers	Papers
Test Cases	**10**	[29], [30], [32], [35]–[38], [40]–[42]
Test Case Generation	7	[29], [30], [32], [35], [37], [40], [41]
Test Case Classification	1	[42]
Test Case Improvement	1	[38]
Natural Language Test Cases Clustering	1	[36]
Requirements Analysis	**6**	[31], [33], [35], [40], [43], [44]
Test Reports	**5**	[45]–[49]
Test Reports Prioritization	2	[46], [49]
Test Reports Augmentation	1	[45]
Test Reports Clustering	1	[47]
Test Reports Quality Assessment	1	[48]
Bug Reports	**2**	[50], [51]
Bug Report Analysis	1	[51]
Bug Reports Classification	1	[50]
Defect Prediction	**2**	[51], [52]
Test Oracle Generation	**2**	[31], [43]
Fault Injection and Seeding	**1**	[34]
Flaky Tests Prediction	**1**	[53]
Vulnerability Prediction	**1**	[51]
Acceptance Test Generation	**1**	[35]

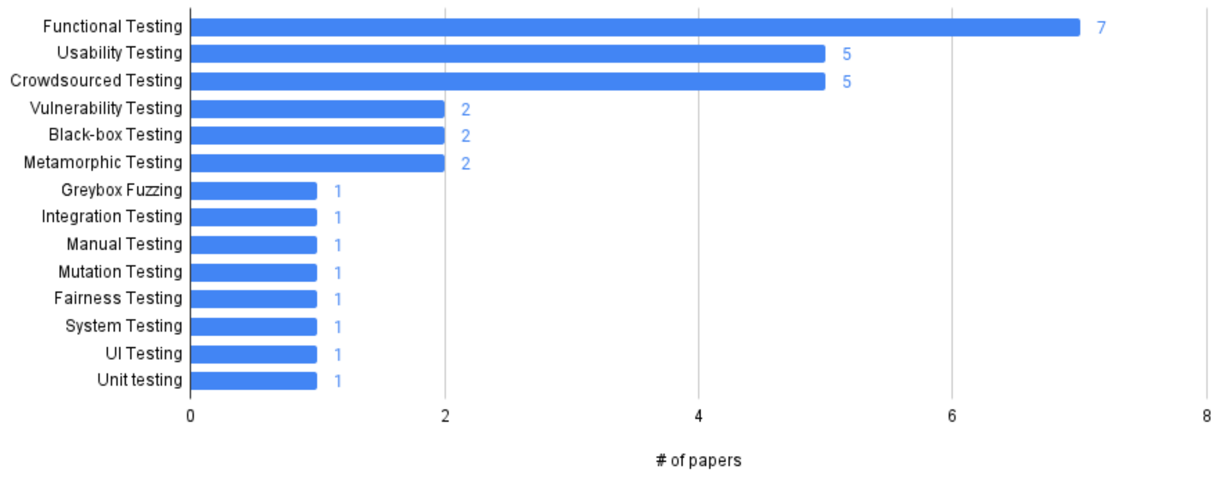


FIGURE 8. Software testing types.

incorporate source code from programming languages. This limitation in language support suggests poor support from the NLP tools and/or abundant resources in the English and Chinese languages. The integration of source code embraces its importance in software testing either the code of SUT itself or the source code of the test cases.

D. RQ4: WHAT ARE THE TRENDING TESTING CONTEXTS/TYPES?

This section delves into the world oif testing as revealed by the research papers under scrutiny. The examination encompasses a spectrum of testing contexts and types that researchers have explored.

The testing type represents a specific category or classification of testing activities based on the testing objectives, focus, and the nature of the defects being addressed. Testing types help define the purpose and scope of the testing effort within the broader testing context.

The testing context refers to the effective area of the software life-cycle under which software testing is conducted or used to improve software testing. It encompasses the broader factors that influence the testing strategy, methodologies, and tools chosen for a particular testing effort.

As shown in Figure 7 and Table 3, the distribution of software testing contexts across the studied research papers reflects a diverse array of testing methodologies and objectives. Test case generation and requirements analysis are prominent, with 7 and 6 instances, respectively, showcasing a significant focus on creating effective test cases and analyzing system requirements. Other contexts, such as test case classification, test case improvement, and Natural language test case clustering, are explored in a more limited manner, each appearing in 1 research paper. The analysis extends to

TABLE 4. Software testing types.

Software Testing Type	Number of papers	Papers
Functional Testing	7	[29], [33], [35], [40], [41], [43], [44]
Usability Testing	5	[45]–[49]
Crowdsourced Testing	5	[45]–[49]
Vulnerability Testing	2	[37], [51]
Black-box Testing	2	[32], [53]
Metamorphic Testing	2	[31], [32]
Greybox Fuzzing	1	[51]
Integration Testing	1	[42]
Manual Testing	1	[36]
Mutation Testing	1	[34]
Fairness Testing	1	[30]
System Testing	1	[52]
UI Testing	1	[41]
Unit Testing	1	[38]

TABLE 5. Machine learning and deep learning techniques.

ML/DL Techniques	# of papers	Papers
Machine Learning Techniques	**7**	**[36]–[38], [42], [47], [48], [50]**
Logistic Regression	3	[38], [48], [50]
Random Forests	2	[38], [50]
Support Vector Machine	2	[38], [50]
Naive Bayes	2	[37], [50]
Hierarchical Agglomerative Clustering (HAC)	2	[36], [47]
K-means	1	[36]
Decision Tree	1	[38]
IFROWANN [54]	1	[42]
Deep Learning	**3**	[38], [51], [53]
Feedforward Neural Network	1	[53]
Back-propagation Neural Network	1	[38]
FuzzGuard [55]	1	[51]
Bidirectional LSTM (BiLSTM)	1	[35]
Activation Functions and Optimizers	**2**	[35], [53]
Softmax	2	[35], [53]
Sigmoid	1	[35]
Adam Optimizer	1	[53]
Computer Vision Techniques	**2**	[41], [47]
Spatial Pyramid Matching (SPM)	1	[47]
Bilateral Filtering	1	[41]
Averaging	1	[41]
Other methods	**2**	[37], [38]
K-fold	1	[38]
Expectation Maximization	1	[37]

various phases of testing, including test report prioritization, test report clustering, and test report quality assessment, each represented in 2 or fewer papers. Additionally, the research delves into specialized areas such as fault injection, fault seeding, bug report analysis, bug report classification, defect prediction, and test oracle generation, each appearing in 2 or fewer papers. Unique testing contexts, like Flaky test prediction, vulnerability prediction, and acceptance test generation, are also explored in 1 research paper each. This distribution shows the diversified nature of software testing.

Figure 8 and Table 4 illustrate the software testing types utilized in the studied research papers, Functional Testing emerges as the predominant type, present in 9 papers. Usability Testing follows closely, appearing in 5 papers. Vulnerability Testing is explored in 3 papers, while Black-box testing, crowdsourced testing, and metamorphic testing each make an appearance in 2 papers. Greybox fuzzing, integration testing, manual testing, mutation testing, fairness testing, system testing, UI testing, and unit testing are each featured once in the research papers, indicating a diverse range of testing types examined within the literature.

E. RQ5: WHAT ARE THE TRENDING MACHINE AND DEEP LEARNING ALGORITHMS USED ALONGSIDE NLP?

The convergence of machine and deep learning techniques within the realm of NLP holds paramount importance for advancing the capabilities and performance of language-centric applications [56]. This amalgamation enables a more comprehensive understanding of linguistic nuances, semantic relationships, and contextual intricacies. The incorporation of diverse techniques further enriches the toolkit available for tackling NLP challenges [57].

TABLE 6. Software, Tools, Frameworks used.

Software, Tools, Frameworks,...	# of papers	Papers
Programming Languages		
Python	11	[29], [30], [32], [33], [35], [36], [38], [41], [47], [50], [51]
Java	7	[31], [38], [40], [43], [45], [49], [53]
R	1	[37]
NLP Tools, Framworks and Libraries		
BERT	5	[30], [35], [36], [44], [52]
Stanford CoreNLP	3	[31], [37], [45]
Word2Vec	3	[36], [38], [46]
NLTK Package	2	[33], [50]
SpaCy	2	[32], [35]
Jieba Library	2	[46], [47]
Language Technology Platform (LTP)	2	[45], [49]
CodeBERT	1	[53]
µBERT	1	[34]
RoBERTa, WordPiece, DistilBERT	1	[35]
SBERT	1	[36]
PyTorch, Doc2Vec	1	[42]
General Language Understanding Evaluation (GLUE)	1	[44]
Berkeley Neural Parser, UniLM	1	[32]
nlpaug	1	[50]
IKAnalyzer	1	[48]
GATE [58]	1	[29]
Software Testing Tools, Framworks and Libraries		
JUnit	1	[38]
Selenium, Pytest	1	[41]
AFL++	1	[51]
PiTest, DeepMutation [59]	1	[34]
T-VEC [1]	1	[40]
Machine/Deep Learning and Computer Vision Libraries and Tools		
Pandas, Scikit-Learn	1	[50]
Tesseract, YoloV5	1	[41]
XGBoost	1	[38]
KEEL	1	[42]
Other Software Engineering Tools		
IBM DOORS	2	[29], [42]
Eclipse JDT Core Component	2	[38], [53]
Eclipse Papyrus, IBM Rational Rhapsody	1	[29]
Soot [2], FlowDroid [3], XOM [4]	1	[37]
Networkx Library	1	[51]
IBIR [60]	1	[34]
Integrated Development Environments (IDEs)		
Eclipse	3	[29], [45], [49]
Netbeans	1	[43]

Table 5 outlines the machine and deep learning techniques employed in the studied research papers, categorizing them into distinct subcategories. Under "Machine Learning", logistic regression is utilized in 3 papers, random forests, and support vector machine in 2 papers each, while Naive Bayes, hierarchical agglomerative clustering (HAC) [61], K-means, decision tree, and IFROWANN [54] are each applied in 1 paper. In the "Deep Learning" category, the feedforward neural network, back-propagation neural network, and FuzzGuard [55] are featured in 1 paper each. The "Activation Functions and Optimizers" category includes softmax in 2 papers, and Adam optimizer and sigmoid in 1 paper each. Image techniques involve spatial pyramid matching (SPM), bilateral filtering, and averaging, each employed in 1 paper. Lastly, under "Other methods", K-fold [62] and expectation maximization are applied in 1 paper each. This breakdown offers an overview of the diverse array of machine and deep learning techniques explored within the research papers.

F. RQ6: WHAT ARE THE TOOLS AND FRAMEWORKS USED IN THIS FIELD ?

Table 6 enumerates the utilization of various software, tools, libraries, and frameworks in the studied research papers, categorized across different domains. About programming languages, Python is prominently featured in 11 papers, followed by Java in 7 papers, and R in 1 paper. NLP tools, frameworks, and libraries exhibit a diverse spectrum, with BERT appearing in 5 papers, Stanford CoreNLP and Word2Vec in 3 papers each, and several others, including NLTK, SpaCy, Jieba, and CodeBERT, making appearances in 2 papers each. Software testing tools and frameworks, such as JUnit, Selenium, Pytest, AFL++, PiTest, and others, are each employed in 1 paper. Machine/deep learning and

TABLE 7. **Precision, Recall, and F-score of the methods/approaches.**

Precision	Recall	F-score	Paper
99.00%	96.00%	97.48%	[29]
96.72%	96.33%	96.83%	[50]
100.00%	81.80%	89.99%	[37]
92.47%	79.93%	85.74%	[36]
84.17%	87.50%	85.58%	[35]
92.00%	74.00%	82.02%	[38]
81.50%	86.50%	81.00%	[53]
85.18%	75.87%	80.01%	[48]
76.00%	67.00%	80.00%	[44]
83.58%	77.76%	78.72%	[45]
91.00%	69.00%	78.00%	[31]
71.00%	57.00%	64.00%	[52]
95.52%	N/A	N/A	[43]

computer vision libraries/tools, such as Pandas, Scikit-Learn, Tesseract, and YoloV5, are utilized in 1 paper each. Other software engineering tools cover a range of applications, with IBM DOORS and Eclipse JDT Core Component being used in 2 papers each, while Eclipse Papyrus, IBM Rational Rhapsody, Soot, FlowDroid, XOM, Networkx Library, and IBIR appear in 1 paper each. Integrated Development Environments (IDEs) feature Eclipse in 3 papers and Netbeans in 1 paper.

G. RQ7: HOW RELIABLE ARE THE METHODS/ APPROACHES?

The evaluation and benchmarking of NLP-based software testing approaches and methods are paramount to assess the effectiveness in addressing specific challenges [63]. This section meticulously examines the precision, recall, and F-score metrics associated with the diverse array of approaches presented in the studied papers. These metrics serve as quantitative indicators of the performance and efficacy of each method, providing insights into their ability to accurately identify relevant information, minimize false positives and negatives, and strike a balance between precision and recall [64]. These numbers are extracted from the studied papers. When multiple numbers are presented we use the average. When the f-score is not presented we automatically calculate using this formula: $2 \times \frac{(precision \times recall)}{(precision + recall)}$.

We automatically calculated the f-score for 3 papers: [29], [37], [38]. Only 13 papers from the 24 papers have shared the metrics from the evaluation phase of their study. Those numbers are presented in Table 7, sorted by the f-score.

As seen in Figure 9, 6 papers have more than 90% in precision and 3 of those papers have also more than 90% in f-score which shows a high level of reliability and opens the door for industry applications. 6 other papers have more than 80% in f-score, which is an important milestone in NLP-based software testing but still far from what the industry is demanding – near 100% reliability.

H. RQ8: WHAT ARE THE MOST USED DATASETS?

Publicly available datasets are essential for the advancement of NLP. They serve as standardized benchmarks, enabling researchers to evaluate and compare the performance of NLP models. Access to these datasets promotes reproducibility, collaboration, and innovation, acting as foundational tools for developing robust NLP applications [65]. In this section, we are interested in discovering the datasets used by the papers in different phases of the creation of their approaches/tools. We are limited to publicly available datasets, and we omit any private or unavailable datasets mentioned in the papers.

As shown in Table 8, WordNet, VerbNet, PropBank, Multi-Genre Natural Language, and C4 are employed during the NLP Model Development phase. NeuroDebug-2 and CiRA (Conditionals in Requirements Artifacts) are the results of the papers. Defects4J, IDoFT, FlakeFlagger, WINOGENDER, WebQuestionSP, ComplexWebQ, BoolQ, NatQA, and SQuAD are predominantly featured in the Evaluation Phase across different studies.

I. RQ9: WHAT ARE THE RECURRENT OPEN CHALLENGES?

Recurrent open challenges persist in the continuous evolution of NLP, posing avenues for exploration and improvement. As NLP continues to unfold as a transformative force in various applications, addressing persistent challenges remains crucial for unlocking the full potential of language-driven technologies [66].

From evaluating the papers presented in this review, many challenges have popped up many times. We assembled them in this list:

- Generalization across fields and languages: Particularly in requirement analysis papers, it lies in achieving robust generalization. Researchers struggle with extending their approaches and methods to diverse fields and languages, aiming for broad applicability beyond specific contexts.
- Ambiguity in natural language requirements: The persistent challenge of resolving ambiguities within natural language requirements remains a focal point. The vagueness and imprecision in linguistic expressions pose a continuous hurdle, demanding innovative NLP strategies to enhance clarity and precision.
- Exploration of different pre-trained NLP models: Researchers face the challenge of determining which model or combination of models yields the best results, necessitating a comprehensive exploration beyond conventional choices.
- Enhancing the evaluation with multiple datasets: Improving the evaluation of the approaches stands as an ongoing challenge that needs rigorous testing against multiple datasets, ensuring a more comprehensive assessment of their performance.
- Integration of ML/DL phases for improved results: A recurring query revolves around the potential benefits of incorporating machine learning (ML) or deep learning (DL) phases into NLP-based software testing methodologies. The investigation of whether the inclusion of these phases will enhance the overall effectiveness

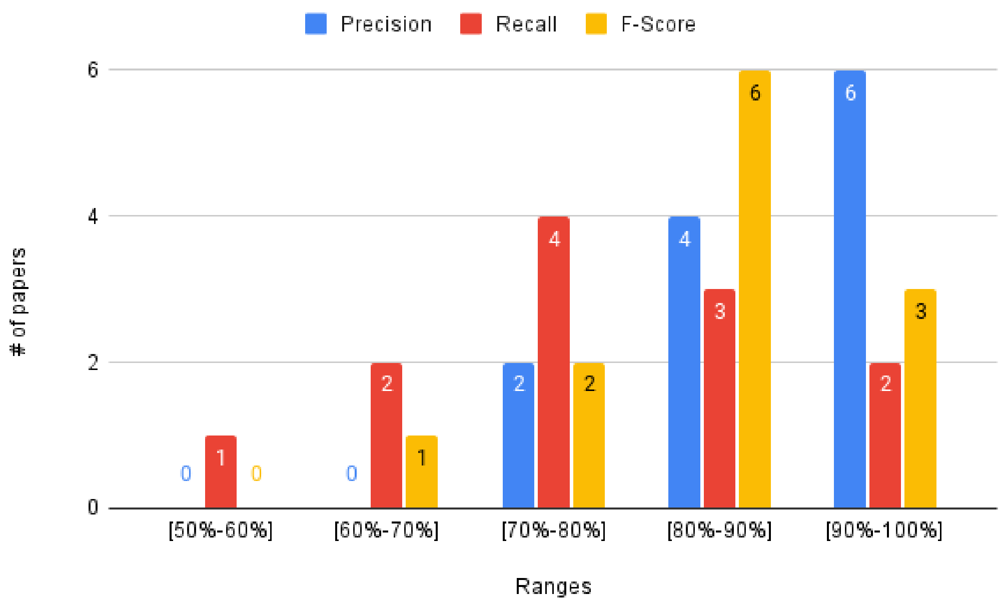


FIGURE 9. Precision, recall, and F-score of the methods/approaches.

TABLE 8. Datasets used in the papers.

Dataset	Phase of Usage	Papers
WordNet	NLP Model Development	[33]
WordNet	NLP Model Development	[29]
VerbNet	NLP Model Development	[29]
PropBank	NLP Model Development	[29]
Multi-Genre Natural Language [5]	NLP Model Development	[44]
C4 (Colossal Clean Crawled Corpus)	NLP Model Development	[41]
NeuroDebug-2 [6]	Release Phase	[52]
CiRA (Conditionals in Requirements Artifacts) [7]	Release Phase	[35]
Defects4J	Evaluation Phase	[34]
IDoFT [8]	Evaluation Phase	[53]
FlakeFlagger [9]	Evaluation Phase	[53]
WINOGENDER [10]	Evaluation Phase	[30]
WebQuestionSP [11]	Evaluation Phase	[32]
ComplexWebQ [12]	Evaluation Phase	[32]
BoolQ [13]	Evaluation Phase	[32]
NatQA [14]	Evaluation Phase	[32]
SQuAD [15]	Evaluation Phase	[32]

and performance of the methods is still an ongoing challenge.

- Precision and Recall Improvement: As we've seen in Table 7, many papers don't have any evaluation of their approach or don't share their evaluation results, and many others are still far from results that are applicable to the market need. The perpetual challenge of enhancing precision and recall metrics remains at the forefront of the field.
- Exploration of different hyperparameters: The exploration of diverse hyperparameters poses an open challenge in the quest for optimal performance. Identifying whether adjusting hyperparameters can yield improvements in the efficacy of the models remains an open question in many papers.
- Transformation into practical tools: Converting innovative methods and approaches into practical tools presents a challenge. Researchers need to bridge the gap between theoretical advancements and real-world applications, necessitating engineering efforts to make their methods readily applicable within the software testing field.

These open challenges pave the road for future research in the field. Whether caused by dataset limitations, engineering constraints, or broader application contexts, these challenges serve as beacons guiding the research community toward uncharted territories of improvement and innovation in the field.

J. RQ10: WHAT ARE THE DOMAINS OF THE SUT?

The domain of the System Under Test (SUT) in software testing is crucial for tailoring testing approaches to specific contexts and requirements. It ensures that testing strategies align with industry standards, regulatory compliance, and user expectations within a particular field [67]. This review

TABLE 9. Domains of the SUT.

Domain	# of papers	Papers
Entertainment (Music, Reading, Photo Sharing, Games, LifeStyle, Food & Drink, Travel)	7	[36], [43], [45]–[49]
Education	6	[43], [45]–[49]
E-commerce	5	[41], [43], [45], [48], [49]
Java Source Code	4	[31], [34], [38], [53]
Embedded Systems	3	[29], [40], [45]
Health	3	[33], [43], [47]
Automotive	2	[35], [40]
Machine Learning Model	1	[30]
Bug Reporting	1	[50]
Software Compilation	1	[52]
Question Anserwing	1	[32]
Software Binary Code	1	[51]
Transportation	1	[42]
Insurance	1	[35]
Finance	1	[46]
Web Conferencing	1	[33]
Telecommunications	1	[35]

question serves as a gateway to understanding the diverse application contexts of the field. We aim to find which domains get the most attention from the researchers.

These domains represent the domain of the SUT used for the evaluation of the suggested approaches/methods in the studied papers.

Table 9 lists the domains of the system under test (SUT) and their respective occurrences in the studied research papers. The most frequently encountered domain is Entertainment (encompassing Music, Reading, Photo Sharing, Games, Lifestyle, Food & Drink, and Travel), appearing in 7 research papers. Education follows closely with 6 occurrences, while E-commerce is associated with 5 research papers. Java source code is mentioned in 4 papers, and embedded systems and health each have 3 occurrences. Automotive is linked to 2 papers, while machine learning model, bug reporting, software compilation, question answering, software binary code, transportation, insurance, finance, web conferencing, and telecommunication are each associated with 1 research paper.

V. DISCUSSION

This paper explores 24 research studies from the field of NLP-based software testing. Most of the studied papers were released in the last 5 years (2023-2019) which may be attributed to the rapidly evolving landscape of NLP and software testing. Advances in NLP techniques, coupled with the increasing integration of NLP in software testing practices, can explain the research output increase during this timeframe.

IEEE, Springer, and Elsevier are the dominant publishers for more than 80% of the papers, this might be attributed to the well-established reputation and widespread recognition of these publishing houses within the academic community. Other publishers may have a smaller share due to variations in the visibility, impact factor, or specialization of different journals.

Tokenization, POS tagging, and TF-IDF being the most used NLP techniques may originate from their fundamental roles in linguistic analysis, information retrieval, and feature representation. And since a good number of papers study natural language text, it is normal for these techniques to pop up.

Only eight papers are sharing in-depth details about their implementation of the method/approach, this limited sharing may be influenced by space constraints in publications or the researchers' discretion. Comprehensive implementation details might be deemed proprietary or too extensive for the scope of the papers, but on the other hand, it makes it hard to reproduce the results of the paper.

The exclusive focus on a single natural language in all papers could be influenced by the complexities and challenges associated with multilingual NLP. Researchers may choose a single language to maintain precision and clarity in their methodology and evaluation. On the other hand, this generates questions about the generalization of the algorithm to other languages. English and Chinese are the most used languages for testing NLP algorithms may reflect the global influence of research conducted in these languages, but the availability of extensive resources and datasets for English and Chinese might also contribute to their popularity. While the absence of implementations in other languages draws the path for future research.

Test case generation and requirements analysis are the most used software testing contexts in more than 50% of the papers. This dominance could be attributed to their critical roles in software development. These contexts directly align with the core objectives of ensuring functionality and meeting specified requirements. Future research should also focus on other software contexts such as defect prediction, and vulnerability prediction. Functional testing, usability testing, and crowdsourced testing are the most used software testing types in more than 70% of the papers. They align with the software testing contexts in their focus on ensuring functional and non-functional requirements. It also shows the focus on crowdsourced testing with all the benefits it has in terms of lowering the cost and leveraging crowd intelligence in software testing.

The integration of machine learning and deep learning alongside NLP in 11 papers is the result of the potential advantage of their union. This combination enhances the ability to handle complex linguistic patterns and solve tasks that cannot be solved with NLP, such as classification and clustering. The choice of logistic regression, random forests, support vector machines, and Naive Bayes in most papers is influenced by their versatility, ease of implementation, and well-established track record in various applications. The popularity of neural networks and LSTM in deep learning may stem from their capacity to capture intricate patterns in data, making them suitable for tasks that require specific linguistic understanding. Still, with less than 50% of the papers, future research should take advantage of this combination to enhance software testing capabilities.

Python and Java are the two trending programming languages for the implementation of the method/approach presented in the studied papers. This could be explained by the large libraries and frameworks that are available in these languages, which support effective development and a vibrant developer community. The popularity of BERT, Stanford CoreNLP, and Word2Vec is linked to their state-of-the-art performance and wide adoption in the NLP community. These libraries offer pre-trained models and robust functionalities. The use of Pandas and Scikit-Learn highlights their adaptability to tasks involving data manipulation and machine learning, which makes them appropriate for a range of research-related applications. JUnit, Selenium, and PyTest were used in two papers indicating that most methods/approaches are not related to the code implementation phase and stick to previous representations in the software engineering lifecycle.

The achievement of high precision and F-score in some papers could be attributed to meticulous algorithm design, effective feature representation, and rigorous evaluation methodologies. These results may showcase the robustness and reliability of specific approaches. But we're still far from the industry need, which requires nearly 0% error rate. We cannot forget to mention that more than 45% of the papers haven't shared the results of their evaluation phase, which can be an alarming number, that impacts the reliability of their results. This adds to the limited sharing of the testing datasets and the source code of the research output. Only 10 papers have open-sourced their implementation and publicly stated the datasets used for the evaluation. This limited sharing can be influenced by factors such as data privacy concerns, the proprietary nature of datasets, or the complexity of releasing software implementations. Researchers might be cautious about sharing sensitive information or investing additional resources in open-sourcing efforts.

There were 17 different umbrella domains of the SUT. This diversity of domains shows the broad applicability of NLP-based software testing across various industries. It also highlights the adaptability of NLP methodologies to different application contexts.

VI. LIMITATIONS AND FUTURE WORK

To ensure the replicability of this review, we meticulously defined and documented the search engines, search terms, and inclusion/exclusion criteria as presented in Section III-B. Two notable challenges in the selection process were the constraints posed by search terms and search engines, as well as potential biases in applying exclusion and inclusion criteria. To address this, we adopted a systematic search approach, employing carefully crafted keywords. Additionally, a manual verification encompassed the examination of reference lists in the initial pool to enhance the likelihood of encompassing all pertinent studies.

Future work should consider solving the challenges presented in Section IV-I. The identification of those open challenges signals an ongoing and dynamic landscape that demands further exploration and innovation. Achieving effective generalization across diverse fields and languages necessitates developing approaches that adapt to varying industry sectors and linguistic contexts. Resolving ambiguities in natural language requirements calls for refining NLP-based testing methods to interpret imprecise specifications accurately, potentially through advanced algorithms and contextual knowledge integration, or through the work on an intermediate representation of those requirements. Higher precision and recall are vital for industry needs, prompting future research to prioritize enhancements, exploring innovative algorithms, optimization techniques, and the integration of machine/deep learning alongside NLP techniques. These research directions aim to advance methodologies and contribute to the development of more powerful testing solutions that meet the diverse needs of different fields and industries. The solution to these open challenges can lead to a better, high-quality, reliable, and lower-cost software that aligns with agile principles

VII. CONCLUSION

In conclusion, this systematic literature review has presented a thorough synthesis of the current state of research concerning NLP-based software testing. We aim to ensure the replicability and rigor of this SLR by establishing clear inclusion/exclusion criteria. Challenges encountered, such as limitations in search terms and engines, were addressed through a systematic approach involving meticulous selection of keywords and manual verifications. The findings underscore the dynamic nature of NLP-based software testing, revealing persistent challenges in generalization across fields and languages, ambiguity in natural language requirements, and the need for precision and recall enhancements to meet industry demands. Despite these obstacles, the literature indicates promising advancements in integrating NLP techniques across diverse software testing scenarios. Notably, the focus on test case generation and requirements analysis reflects current trends, while opportunities for further exploration in less-represented domains remain. Additionally, certain NLP techniques and programming languages exhibit dominance, suggesting discernible trends in the field. As the field progresses, this systematic review serves as a foundational resource for both scholars and industry practitioners, facilitating the identification of future research avenues and improvements, fostering interdisciplinary collaboration, and contributing to the evolving landscape of natural language processing applications in software testing.

REFERENCES

[1] G. J. Myers, T. Badgett, T. M. Thomas, and C. Sandler, *The Art of Software Testing*, vol. 2. Hoboken, NJ, USA: Wiley, 2004.

[2] M. Boukhlif, M. Hanine, and N. Kharmoum, ''A decade of intelligent software testing research: A bibliometric analysis,'' *Electronics*, vol. 12, no. 9, p. 2109, May 2023.

[3] A. Bertolino, ''Software testing research: Achievements, challenges, dreams,'' *Future Softw. Eng.*, pp. 85–103, 2007.

[4] R. Ramler, S. Biffl, and P. Grünbacher, *Value-Based Management of Software Testing*. Springer, 2006, pp. 225–244.

[5] V. Garousi, S. Bauer, and M. Felderer, ''NLP-assisted software testing: A systematic mapping of the literature,'' *Inf. Softw. Technol.*, vol. 126, Oct. 2020, Art. no. 106321.

[6] I. Ahsan, W. H. Butt, M. A. Ahmed, and M. W. Anwar, ''A comprehensive investigation of natural language processing techniques and tools to generate automated test cases,'' in *Proc. 2nd Int. Conf. Internet Things, Data Cloud Comput.*, Mar. 2017, pp. 1–10.

[7] M. J. Escalona, J. J. Gutierrez, M. Mejías, G. Aragón, I. Ramos, J. Torres, and F. J. Domínguez, ''An overview on test generation from functional requirements,'' *J. Syst. Softw.*, vol. 84, no. 8, pp. 1379–1393, Aug. 2011.

[8] F. Nazir, W. H. Butt, M. W. Anwar, and M. A. K. Khattak, ''The applications of natural language processing (NLP) for software requirement engineering—A systematic literature review,'' in *Information Science and Applications*, K. Kim and N. Joukov, Eds. Singapore: Springer, 2017, pp. 85–493.

[9] Z. Aoujil, M. Hanine, E. S. Flores, M. A. Samad, and I. Ashraf, ''Artificial intelligence and behavioral economics: A bibliographic analysis of research field,'' *IEEE Access*, vol. 11, pp. 139367–139394, 2023.

[10] M. Boukhlif, N. Kharmoum, M. Hanine, C. Elasri, W. Rhalem, and M. Ezziyyani, ''Exploring the application of classical and intelligent software testing in medicine: A literature review,'' in *Proc. Int. Conf. Adv. Intell. Syst. Sustain. Develop.*, 2024, pp. 37–46.

[11] S. Tahvili and L. Hatvani, *Artificial Intelligence Methods for Optimization of the Software Testing Process: With Practical Examples and Exercises*. Amsterdam, The Netherlands: Elsevier, 2022.

[12] M. A. Job, ''Automating and optimizing software testing using artificial intelligence techniques,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 5, pp. 594–602, 2021.

[13] H. Alrumaih, A. Mirza, and H. Alsalamah, ''Toward automated software requirements classification,'' in *Proc. 21st Saudi Comput. Soc. Nat. Comput. Conf. (NCC)*, Apr. 2018, pp. 1–6.

[14] C. Elasri, N. Kharmoum, F. Saoiabi, M. Boukhlif, S. Ziti, and W. Rhalem, ''Applying graph theory to enhance software testing in medical applications: A comparative study,'' in *Proc. Int. Conf. Adv. Intell. Syst. Sustain. Develop.*, 2023, pp. 70–78.

[15] A. Kao and S. R. Poteet, *Natural Language Processing and Text Mining*. Cham, Switzerland: Springer, 2007.

[16] W. Zheng, Y. Bai, and H. Che, ''A computer-assisted instructional method based on machine learning in software testing class,'' *Comput. Appl. Eng. Educ.*, vol. 26, no. 5, pp. 1150–1158, Sep. 2018.

[17] R. S. Wahono, ''A systematic literature review of software defect prediction,'' *J. Softw. Eng.*, vol. 1, no. 1, pp. 1–16, 2015.

[18] C. Chao, Q. Yang, and X. Tu, ''Research on test case generation method of airborne software based on NLP,'' in *Proc. CEUR-WS*, vol. 3304, 2022, pp. 28–37.

[19] S. D. R. Konreddy ''The impact of NLP on software testing,'' *J. Univ. Shanghai Sci. Technol.*, vol. 23, no. 8, pp. 295–304, Aug. 2021.

[20] W. Ke, C. Wu, X. Fu, C. Gao, and Y. Song, ''Interpretable test case recommendation based on knowledge graph,'' in *Proc. IEEE 20th Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Dec. 2020, pp. 489–496.

[21] A. Carrera-Rivera, W. Ochoa, F. Larrinaga, and G. Lasa, ''How-to conduct a systematic literature review: A quick guide for computer science research,'' *MethodsX*, vol. 9, Nov. 2022, Art. no. 101895.

[22] K. Petersen, S. Vakkalanka, and L. Kuzniarz, ''Guidelines for conducting systematic mapping studies in software engineering: An update,'' *Inf. Softw. Technol.*, vol. 64, pp. 1–18, Aug. 2015.

[23] A. Rai and S. Borah, ''Study of various methods for tokenization,'' in *Applications of Internet of Things*. Cham, Switzerland: Springer, 2021, pp. 193–200.

[24] I. Lauriola, A. Lavelli, and F. Aiolli, ''An introduction to deep learning in natural language processing: Models, techniques, and tools,'' *Neurocomputing*, vol. 470, pp. 443–456, Jan. 2022.

[25] J. Zhang, F. Meng, M. Wang, D. Zheng, W. Jiang, and Q. Liu, ''Is local window essential for neural network based Chinese word segmentation?'' in *Lecture Notes in Computer Science*. Cham, Switzerland: Springer, 2016, pp. 450–457.

[26] N. Z. Abdurakhmonova, A. S. Ismailov, and D. Mengliev, ''Developing NLP tool for linguistic analysis of turkic languages,'' in *Proc. IEEE Int. Multi-Conf. Eng., Comput. Inf. Sci. (SIBIRCON)*, Nov. 2022, pp. 1790–1793.

[27] P. Sun, X. Yang, X. Zhao, and Z. Wang, ''An overview of named entity recognition,'' in *Proc. Int. Conf. Asian Lang. Process. (IALP)*, Nov. 2018, pp. 273–278.

[28] Y. Ling, Y. An, M. Liu, S. A. Hasan, Y. Fan, and X. Hu, ''Integrating extra knowledge into word embedding models for biomedical NLP tasks,'' in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 968–975.

[29] C. Wang, F. Pastore, A. Goknil, and L. C. Briand, ''Automatic generation of acceptance test cases from use case specifications: An NLP-based approach,'' *IEEE Trans. Softw. Eng.*, vol. 48, no. 2, pp. 585–616, Feb. 2022.

[30] E. Soremekun, S. Udeshi, and S. Chattopadhyay, ''Astraea: Grammar-based fairness testing,'' *IEEE Trans. Softw. Eng.*, vol. 48, no. 12, pp. 5188–5211, Dec. 2022.

[31] A. Blasi, A. Gorla, M. D. Ernst, M. Pezzè, and A. Carzaniga, ''MeMo: Automatically identifying metamorphic relations in javadoc comments for test automation,'' *J. Syst. Softw.*, vol. 181, Nov. 2021, Art. no. 111041.

[32] X. Xie, S. Jin, and S. Chen, ''QAASKER$^+$: A novel testing method for question answering software via asking recursive questions,'' *Automated Softw. Eng.*, vol. 30, no. 1, p. 14, May 2023.

[33] Z. Peng, P. Rathod, N. Niu, T. Bhowmik, H. Liu, L. Shi, and Z. Jin, ''Testing software's changing features with environment-driven abstraction identification,'' *Requirements Eng.*, vol. 27, no. 4, pp. 405–427, Dec. 2022.

[34] M. Ojdanic, A. Garg, A. Khanfir, R. Degiovanni, M. Papadakis, and Y. Le Traon, ''Syntactic versus semantic similarity of artificial and real faults in mutation testing studies,'' *IEEE Trans. Softw. Eng.*, vol. 49, no. 7, pp. 3922–3938, Jul. 2023.

[35] J. Fischbach, J. Frattini, A. Vogelsang, D. Mendez, M. Unterkalmsteiner, A. Wehrle, P. R. Henao, P. Yousefi, T. Juricic, J. Radduenz, and C. Wiecher, ''Automatic creation of acceptance tests by extracting conditionals from requirements: NLP approach and case study,'' *J. Syst. Softw.*, vol. 197, Mar. 2023, Art. no. 111549.

[36] M. Viggiato, D. Paas, C. Buzon, and C.-P. Bezemer, ''Identifying similar test cases that are specified in natural language,'' *IEEE Trans. Softw. Eng.*, vol. 49, no. 3, pp. 1027–1043, Mar. 2023.

[37] B. F. Demissie, M. Ceccato, and L. K. Shar, ''Security analysis of permission re-delegation vulnerabilities in Android apps,'' *Empirical Softw. Eng.*, vol. 25, no. 6, pp. 5084–5136, Nov. 2020.

[38] C. Wei, L. Xiao, T. Yu, X. Chen, X. Wang, S. Wong, and A. Clune, ''Automatically tagging the 'AAA' pattern in unit test cases using machine learning models,'' *IEEE Trans. Softw. Eng.*, vol. 49, no. 5, pp. 3305–3324, May 2023.

[39] O. Majewska, I. Vulic, and A. Korhonen, *Linguistically Guided Multilingual NLP: Current Approaches, Challenges, and Future Perspectives*. Boca Raton, FL, USA: CRC Press, 2022.

[40] G. Carvalho, D. Falcão, F. Barros, A. Sampaio, A. Mota, L. Motta, and M. Blackburn, ''NAT2TESTSCR: Test case generation from natural language requirements based on SCR specifications,'' *Sci. Comput. Program.*, vol. 95, pp. 275–297, Dec. 2014.

[41] Z. Khaliq, D. A. Khan, and S. U. Farooq, ''Using deep learning for selenium web UI functional tests: A case-study with e-commerce applications,'' *Eng. Appl. Artif. Intell.*, vol. 117, Jan. 2023, Art. no. 105446.

[42] S. Tahvili, L. Hatvani, E. Ramentol, R. Pimentel, W. Afzal, and F. Herrera, ''A novel methodology to classify test cases using natural language processing and imbalanced learning,'' *Eng. Appl. Artif. Intell.*, vol. 95, Oct. 2020, Art. no. 103878.

[43] M. I. Malik, M. A. Sindhu, A. S. Khattak, R. A. Abbasi, and K. Saleem, "Automating test oracles from restricted natural language agile requirements," *Expert Syst.*, vol. 38, no. 1, pp. 1–12, Jan. 2021.

[44] M. Y. Chow, "Analysis of embedded system's functional requirement using BERT-based name entity recognition for extracting IO entities," *J. Inf. Process.*, vol. 31, no. 1, pp. 143–153, 2023.

[45] X. Chen, H. Jiang, Z. Chen, T. He, and L. Nie, "Automatic test report augmentation to assist crowdsourced testing," *Frontiers Comput. Sci.*, vol. 13, no. 5, pp. 943–959, Oct. 2019.

[46] P. Zhu, Y. Li, T. Li, H. Ren, and X. Sun, "Advanced crowdsourced test report prioritization based on adaptive strategy," *IEEE Access*, vol. 10, pp. 53522–53532, 2022.

[47] D. Liu, Y. Feng, X. Zhang, J. A. Jones, and Z. Chen, "Clustering crowdsourced test reports of mobile applications using image understanding," *IEEE Trans. Softw. Eng.*, vol. 48, no. 4, pp. 1290–1308, Apr. 2022.

[48] X. Chen, H. Jiang, X. Li, L. Nie, D. Yu, T. He, and Z. Chen, "A systemic framework for crowdsourced test report quality assessment," *Empirical Softw. Eng.*, vol. 25, no. 2, pp. 1382–1418, Mar. 2020.

[49] Y. Yang and X. Chen, "Crowdsourced test report prioritization based on text classification," *IEEE Access*, vol. 10, pp. 92692–92705, 2022.

[50] S. A. Alsaedi, A. Y. Noaman, A. A. A. Gad-Elrab, and F. E. Eassa, "Nature-based prediction model of bug reports based on ensemble machine learning model," *IEEE Access*, vol. 11, pp. 63916–63931, 2023.

[51] F. Rustamov, J. Kim, J. Yu, H. Kim, and J. Yun, "BugMiner: Mining the hard-to-reach software vulnerabilities through the target-oriented hybrid fuzzer," *Electronics*, vol. 10, no. 1, p. 62, Dec. 2020.

[52] F. Artuso, G. A. Di Luna, and L. Querzoni, "Debugging debug information with neural networks," *IEEE Access*, vol. 10, pp. 54136–54148, 2022.

[53] S. Fatima, T. A. Ghaleb, and L. Briand, "Flakify: A black-box, language model-based predictor for flaky tests," *IEEE Trans. Softw. Eng.*, vol. 49, no. 4, pp. 1912–1927, Apr. 2023.

[54] E. Ramentol, S. Vluymans, N. Verbiest, Y. Caballero, R. Bello, C. Cornelis, and F. Herrera, "IFROWANN: Imbalanced fuzzy-rough ordered weighted average nearest neighbor classification," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 5, pp. 1622–1637, Oct. 2015.

[55] P. Zong, T. Lv, D. Wang, Z. Deng, R. Liang, and K. Chen, "FuzzGuard: Filtering out unreachable inputs in directed grey-box fuzzing through deep learning," in *Proc. 29th USENIX Secur. Symp.*, Aug. 2020, pp. 2255–2269.

[56] U. Kamath, J. Liu, and J. Whitaker, *Deep Learning for NLP and Speech Recognition*, vol. 84. Cham, Switzerland: Springer, 2019.

[57] A. Celikyilmaz, L. Deng, and D. Hakkani-Tür, *Deep Learning in Spoken and Text-Based Dialog Systems*. Cham, Switzerland: Springer, 2018, pp. 49–78.

[58] H. Cunningham, "Gate: A framework and graphical development environment for robust NLP tools and applications," in *Proc. 40th Annu. meeting Assoc. Comput. linguistics*, 2002, pp. 168–175.

[59] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, and Y. Wang, "DeepMutation: Mutation testing of deep learning systems," in *Proc. IEEE 29th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2018, pp. 100–111.

[60] A. Khanfir, A. Koyuncu, M. Papadakis, M. Cordy, T. F. Bissyandé, J. Klein, and Y. Le Traon, "IBiR : Bug-report-driven fault injection," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 2, pp. 1–31, Mar. 2023.

[61] R. Alfred, D. Kazakov, M. Bartlett, and E. Paskaleva, "Hierarchical agglomerative clustering for cross-language information retrieval," *Int. J. Transl.*, vol. 19, no. 1, pp. 139–162, 2007.

[62] O. Chamorro-Atalaya, J. Arévalo-Tuesta, D. Balarezo-Mares, A. Gonzáles-Pacheco, O. Mendoza-León, M. Quipuscoa-Silvestre, G. Tomás-Quispe, and R. Suarez-Bazalar, "K-fold cross-validation through identification of the opinion classification algorithm for the satisfaction of university students," *Int. J. Online Biomed. Eng.*, vol. 19, no. 11, pp. 1–12, Aug. 2023.

[63] P. Resnik and J. Lin, *Evaluation of NLP Systems*. Hoboken, NJ, USA: Wiley, 2010, ch. 1, pp. 271–295.

[64] T. Ly, C. Pamer, O. Dang, S. Brajovic, S. Haider, T. Botsis, D. Milward, A. Winter, S. Lu, and R. Ball, "Evaluation of natural language processing (NLP) systems to annotate drug product labeling with MedDRA terminology," *J. Biomed. Informat.*, vol. 83, pp. 73–86, Jul. 2018.

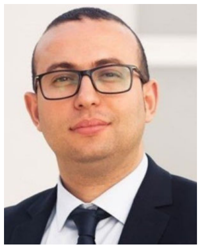
[65] Q. Lhoest et al., "Datasets: A community library for natural language processing," 2021, *arXiv:2109.02846*.

[66] Z. Kaddari, Y. Mellah, J. Berrich, M. G. Belkasmi, and T. Bouchentouf, "Natural language processing: Challenges and future directions," in *Artificial Intelligence and Industrial Applications*. Springer, 2021, pp. 236–246.

[67] T. Martínez-Ruiz, J. Münch, F. García, and M. Piattini, "Requirements and constructors for tailoring software processes: A systematic literature review," *Softw. Qual. J.*, vol. 20, no. 1, pp. 229–260, Mar. 2012.

MOHAMED BOUKHLIF received the Engineering degree (equivalent to the Master of Engineering) from the University of Hassan II, Morocco, with a focus on computer science engineering, big data, and cloud computing. He is currently pursuing the Ph.D. degree with Chouaib Doukkali University, Morocco. In 2024, he was the Head of the IT Department, National School of Applied Sciences, University Cadi Ayyad, Morocco. His research interests include software engineering, software testing, and applied artificial intelligence.

MOHAMED HANINE received the Ph.D. degree in computer science (spatial decision-making) from the University of Cadi Ayyad, Marrakesh, Morocco, in 2017. In 2018, he joined the Department of Telecommunications, Networks, and Computer Science, National School of Applied Sciences, where he educates engineering students in the fields of big data, artificial intelligence, NoSQL, and business intelligence. He is currently an Associate Professor with the National School of Applied Sciences, Chouaib Doukkali University, El Jadida, Morocco. His research interests include artificial intelligence, big data, multicriteria decision-making, NoSQL, and business intelligence.



NASSIM KHARMOUM received the Diploma of Doctor degree in computer science from Mohammed V University in Rabat. He is currently pursuing the Ph.D. degree in software engineering with the National Center for Scientific and Technical Research, Morocco. He proposed different validated methods in scientific articles published in international journals, books, and conferences. His research interests include analysis and conceptual modeling, software requirements, multi-agent systems modeling, and modernization of legacy systems.

ATENEA RUIGÓMEZ NORIEGA is currently with Universidad Europea del Atlántico, Santander, Spain. She is also affiliated with Universidad Internacional Iberoamericana, Campeche, Mexico, and Universidad Internacional Iberoamericana, Arecibo, PR, USA.

DAVID GARCÍA OBESO is currently with Universidad Europea del Atlántico, Santander, Spain. He is also affiliated with Universidade Internacional do Cuanza, Kuito, Bié, Angola, and Universidad de La Romana, La Romana, Dominica.

IMRAN ASHRAF (Member, IEEE) received the M.S. degree (Hons.) in computer science from Blekinge Institute of Technology, Karlskrona, Sweden, in 2010, and the Ph.D. degree in information and communication engineering from Yeungnam University, Gyeongsan, South Korea, in 2018. He was a Postdoctoral Fellow with Yeungnam University. He is currently an Assistant Professor with the Information and Communication Engineering Department, Yeungnam University. His research interests include positioning using next-generation networks, communication in 5G and beyond, location-based services in wireless communication, smart sensors (LIDAR) for smart cars, and data analytics.