ECTI Transactions on Computer and Information Technology
Journal homepage: https://ph01.tci-thaijo.org/index.php/ecticit/
Published by the ECTI Association, Thailand, ISSN: 2286-9131

A Systematic Mapping Review: Tracking the Relationships Between Software Artifacts using NLP

Fedaa khalil[1], Ghaida rebdawi[2] and Nada ghneim[3]

ABSTRACT

In software development, traceability from requirements to realization is essential, yet manual tracing is labour-intensive and prone to errors. Requirements Traceability (RT) is crucial for effective management and impact assessment. This paper explores the application of Natural Language Processing (NLP) in requirements traceability (RT) through a systematic mapping review of literature from 2019 to 2023. Out of 209 initial studies, we selected 49 using stringent criteria. RT approaches were categorised into ontology-based and embedding techniques. Embedding techniques have gained prominence for their ability to capture relationships between artifacts. We identified two primary paradigms in RT: rule-based methods, which use predefined heuristics, and machine learning approaches, including traditional classifiers. Machine learning models significantly improve accuracy and adaptability, especially when paired with advanced embeddings. A notable trend is the increasing reliance on standard datasets, like the CoEST repository, to validate methods, enhance reproducibility, and enable robust comparisons. Despite advancements, challenges persist. Non-functional requirements remain underexplored, and the lack of comprehensive benchmarks limits the generalizability of current approaches. Future research should focus on creating inclusive datasets with diverse requirements and integrating hybrid methods to improve performance. Overall, the study underscores the critical role of embedding techniques in RT while highlighting gaps and opportunities for advancing the field.

Article information:

Keywords: Natural Language Processing, Requirements Traceability, Requirements Representation, Syntax, Semantic

Article history:
Received: October 3, 2024
Revised: March 22, 2025
Accepted: March 29, 2025
Published: April 26, 2025 (Online)

DOI: 10.37936/ecti-cit.2025192.258621

1. INTRODUCTION

In the rapidly evolving landscape of software development, ensuring the traceability of requirements from their initial articulation to their final implementation is critical [1]. Requirements Traceability (RT) is pivotal in various aspects of software engineering, including impact analysis, project management, and quality assurance [2]. Despite its importance, manual tracing of requirements across various software artifacts, such as design documents, source code, and test cases, remains a labour-intensive, time-consuming, and error-prone task. As software systems grow in complexity, the limitations of manual tracing become more pronounced, leading to increased risks of defects and project delays.

Natural Language Processing (NLP) has emerged as a transformative tool for addressing these challenges. Since developers document most software requirements in natural language, NLP techniques are well-suited to automate and enhance RT processes. By leveraging advancements in machine learning (ML) and deep learning (DL), NLP offers the potential to significantly improve traceability accuracy, scalability, and efficiency. Despite its promise, current applications of NLP in RT have yet to fully capitalize on these advancements. The representation of requirements—a key factor in achieving robust and accurate traceability—remains underexplored, and gaps persist in integrating state-of-the-art NLP methods into RT practices.

[1,2]The authors are with the Department of Informatics, Higher Institute for Applied Sciences and Technology (HIAST), Damascus, Syria, Email: fedaa.khalil@hiast.edu.sy and Ghaida.rebdawi@hiast.edu.sy
[3]The author is with the Faculty of Information and Communication Technology, Arab International University (AIU), Damascus, Syria, Email: n-ghneim@aiu.edu.sy
[1]Corresponding author: fedaa.khalil@hiast.edu.sy

A review of existing literature underscores these limitations. Previous studies [3–8] have primarily focused on traditional information retrieval (IR) methods or early ML applications, often overlooking how advanced NLP techniques and innovative requirement representations can enhance RT. Some works [3, 5] emphasize solution methods and ML applications but neglect requirement representation strategies, while others [4, 6, and 7] explore IR and basic NLP techniques. A few studies, such as [8], delve into specific traceability scenarios like issue-commit traceability but lack a broader analysis of modern NLP's potential impact.

This paper aims to fill these gaps by conducting a systematic mapping review of the literature published between 2019 and 2023, focusing specifically on applying advanced NLP techniques in RT. The review encompasses 209 papers, from which we selected 49 based on stringent inclusion and exclusion criteria. Our analysis not only categorises the existing approaches to RT but also highlights emerging trends, identifies gaps in current research, and proposes future directions to address these gaps.

The contributions of this paper are threefold:

Comprehensive Analysis: We provide a detailed overview of state-of-the-art NLP-based RT, highlighting the growing adoption of advanced embedding techniques and standard datasets that enhance reproducibility.

Impact of Representations: We demonstrate the critical influence of requirement representations on traceability accuracy, offering actionable insights into optimizing these representations.

Research Gaps and Future Directions: We identify significant gaps, such as the limited focus on non-functional requirements, adopting a method of representing artifacts according to their nature (natural language or software), and propose targeted research pathways to address these limitations.

By addressing the limited focus on non-functional requirements and developing methods to represent artifacts according to their nature—whether as natural language (e.g., textual requirements) or software (e.g., code, models)—developers can ensure that software can better meet critical quality attributes such as performance, scalability, security, and usability. This dual approach enables developers and teams to establish precise mappings between requirements and downstream artifacts, ensuring that requirements are consistently implemented, tested, and validated throughout the software lifecycle. As a result, issues can be identified earlier in development, streamlining processes, reducing rework, and enhancing both software quality and efficiency.

In summary, this study emphasizes the transformative potential of modern NLP techniques in addressing longstanding challenges in RT. By bridging gaps in current research and providing actionable insights, our work aims to advance the state of software engineering, paving the way for more reliable, efficient, and scalable RT practices.

In this study, we addressed six RQs to provide the current progress of the studies on RT. (RQ stands for "Research Question." In academic research, RQs are specific questions that guide the study and structure the research process. They help researchers focus on particular aspects of the topic they are investigating and aim to answer these questions through their research.)

RQ1: What is the state of the published literature on RT?

RQ2: What are the proposed representations in the literature for RT?

RQ3: What are the proposed solutions in the literature for RT?

RQ4: What are the types of RT in the literature?

RQ5: What are the datasets used in this field?

RQ6: What gaps and potential future directions exist in this field?

This study is structured as follows. First, section 2 provides a background of RT and NLP. Then we present our methodology in detail in section 3. Section 4 describes and explains the results obtained by the analysis. In section 5, we showed threats to validity. We close the report by concluding it in section 5.

2. BACKGROUND

Gotel *et al.* define traceability as "the ability to describe and follow the life of a requirement, in both a forward and backward direction" [5]. Any relationship between artifacts involved in the software engineering life cycle is a path from one artifact to another, and it is called a trace link. RT usually includes planning and managing traceability strategy, generating, refining, maintaining, representing, and using trace links. Traceability can be horizontal traceability, which is traceability between requirements, or vertical traceability, which is traceability between requirements and other artifacts [6].

Conversely, NLP, which is part of Artificial Intelligence (AI), enables computers to understand and process natural language texts to achieve a specific purpose [7].

An NLP-based approach typically operates at three primary levels: lexical, syntactic, and semantic.

-Lexical level: It focuses on word analysis and includes several tasks such as:

Tokenization: splitting a text into a list of tokens, which can be words, numbers, or punctuation marks.

Lemmatization: identifying base or dictionary form (lemma).

-Syntactic level: It focuses on analysing the grammatical structure of sentences and includes several tasks:

Part-of-Speech Tagging (POS-tagging): tagging each token in a sentence with its corresponding part of speech tag based on its syntactical context.

Chunking: detecting syntactic components (Noun Phrases and Verb Phrases).

-Semantic level: It focuses on understanding the meaning of the text by automatically mapping a natural language sentence into a formal representation of its meaning. The literature has proposed different semantic representations:

Ontology: It is a data model representing a set of concepts within a domain and the relationships between those concepts. WordNet is an example of an ontology.

esentation model depicting text as a term-by-document matrix. The Bag-of-Words (BOW) model treats word frequencies as weights and uses words as features, making it a special case of the Vector Space Model (VSM). VSM uses other weighting factors, such as IDF and TF-IDF.

Topic Modelling-Based Representation: It is a statistical modelling approach used to discover the latent or abstract topics in a set of texts, such as Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA).

Embedding: It is a method for learning high-quality vector representations of words from large amounts of unstructured text data, which allows words with similar meanings to have similar vector representations, such as Word2Vec, GloVe, BERT, etc.

3. OUR METHODOLOGY

The goal of this review is to identify, categorise, and analyse existing literature published between 2019 and August 2023 that handled NLP-based approaches to traceability requirements tasks.

3.1 Planning

This section contains the research questions, method, and the acceptance and rejection criteria used to filter the results.

3.1.1 Research Questions

We determined the following main research questions:

RQ1: What is the state of the published literature on RT?

RQ2: What are the proposed representations in the literature for RT?

RQ3: What are the proposed solutions in the literature for RT?

RQ4: What are the types of RT in the literature?

RQ5: What are the datasets used in this field?

RQ6: What gaps and potential future directions exist in this field?

3.1.2 Search Query and Data Sources

We used six digital libraries as the data sources for our mapping study: Scopus, IEEE Xplore, ACM Digital Library, Elsevier, MDPI, and SpringerLink. We chose them as they host the major journals and conference proceedings related to software engineering (SE) and RE.

We used the significant terms “Traceability” (representing the context of the research) and “Natural Language Processing” (representing the intervention in this context) as the base terms, and we also included “Software artifacts”. We enrich each of these terms by adding synonyms and sub-fields. Table 1 shows the whole set of selected keywords for this study divided into three groups: A, B, and C. We used them to create the final search query (A AND B AND C).

Table 1: *Keywords used in our study.*

Set	Term	Derived Keywords
A	Traceability	Requirements tracking, artifacts traceability, Trace link recovery, trace retrieval
B	Natural Language Processing	NLP, statistical NLP, machine learning, deep learning, information extraction, information retrieval, text mining, text analysis, linguistic instruments, linguistic approaches
C	Software artifacts	Source code, tests, documentation, requirements, UML diagrams

3.1.3 Acceptance and rejection criteria

Acceptance and rejection criteria serve as a way to filter out papers that are not related to our research questions.

We accepted research papers that present an approach related to the field of RT, published between 2019 and August 2023.

We rejected the papers that are secondary research (such as literature reviews, summaries, etc.), published in languages other than English, duplicated, or lacking detailed information about the proposed approach (such as short papers, posters, etc.).

3.1.4 Search Procedure

Starting from the defined data sources, we obtained 209 candidate papers. We manually eliminated duplicated papers by comparing authors, titles, and abstracts. After removing all duplicates, 155 papers remained. After rejecting secondary research papers and short research papers, 104 papers remained.

We filtered these papers by applying the selection criteria based on titles and abstracts. This stage led

to the selection of 74 papers. We then conducted a full-text review of those papers to discard documents that did not satisfy our selection criteria. The remaining primary research papers after this stage are 49.

Fig. 1 shows the previously mentioned stages and the number of selected publications after each stage.

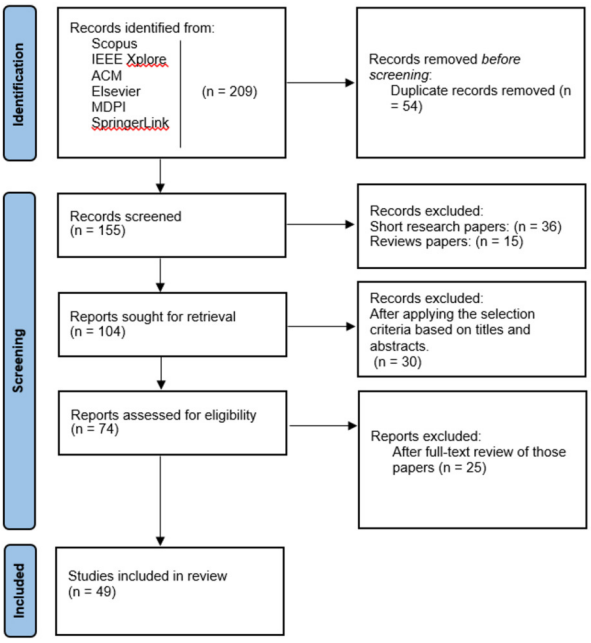

Fig.1: *Overview of steps in mapping study planning.*

4. RESULTS AND DISCUSSION

In this section, we describe and explain the results obtained by analysing our final list, containing 49 unique peer-reviewed research papers, to answer the research questions posed in the previous section.

RQ1: What is the state of the published literature on RT?

The vast majority of papers in our final list (more than 85%) were published between 2019 and 2021. Fig. 2 shows the distribution of the paper publications per year.

The publication of papers in our final list is as indicated below: 51% in IEEE International Conference, 10% in Springer Link, 10% in ACM, 6% in MDPI, 6% in Elsevier, and the remaining in other Scopus-indexed journals.

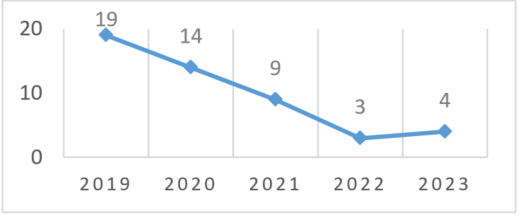

Fig.2: *Number of published papers per year.*

Among the selected papers, 35% appeared in journals, 57% were published in conference proceedings, 6% came from workshops, and one appeared in ACM SIGSOFT Software Engineering Notes. Fig. 3 and 4 show the distribution of published papers per library and type, respectively.

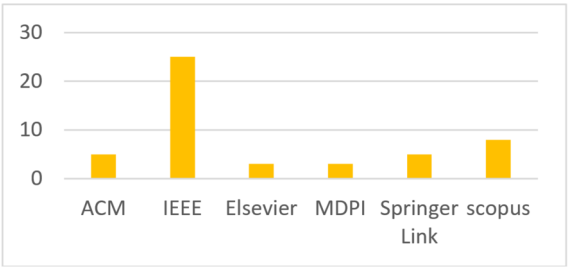

Fig.3: *Number of published papers per year.*

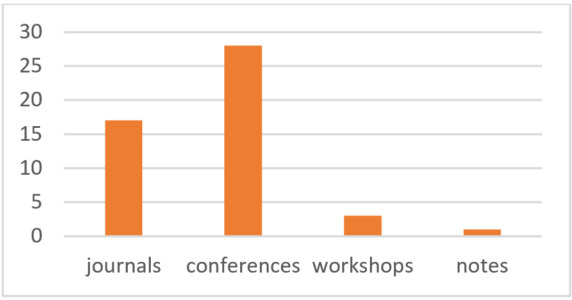

Fig.4: *Number of published papers per year.*

RQ2: What are the proposed representations and solutions in the literature for RT?

We found several types of text representation techniques in this study. We can divide these representation techniques into categories based on the concept: Ontology representation, Lexical and Syntactic features, Probabilistic Models, VSM representation, Topic modelling representation, and Advanced Embedding representation. Fig. 5 shows the hierarchy of papers categorised based on their targeted representation method.

-Ontology Representation: Using this type requires the presence of an ontology to cover every project domain, which can be quite challenging and poses a significant difficulty in applying this form of representation. Few works have used ontology as a representation method. This type comprises 6.1% of papers (3 out of 49).

-Lexical and Syntactic features: While lexical and syntactic features provide information about word order and grammatical relationships, they do not explicitly reveal semantic roles. Strict adherence to syntax may limit the flexibility needed to accommodate changes in requirements since input is a predefined set of linguistic features. This type comprises 14.2% of papers (7 out of 49).

-Probabilistic Models: Probability theory represents uncertainty in the relevance of documents to

Fig.5: *The hierarchy of papers categorised based on their targeted representation method.*

queries. These models rank documents based on a given query and are often more effective than non-probabilistic models. JS (Jensen-Shannon divergence) is an example of these models. This type comprises 10.2% of papers (5 out of 49).

-VSM (Vector Space Model) representation: VSM can handle documents of varying lengths and structures, making it suitable for representing different software requirements. However, it treats words independently without considering their semantic relationships, and longer documents tend to have higher frequencies, which can skew results. VSM treats documents as bags of words, without considering the structural organisation of the text, where input is a vector of words, tokens, or stems combined with their weights. Such an approach may limit its effectiveness in capturing hierarchical relationships or dependencies within requirements. This type comprises 36.7% of papers (18 out of 49) that use TF-IDF. Some of the works combine the BOW technique with TF-IDF (3 out of 49). One paper uses BOW.

-Topic modelling representation: Topic modelling extracts hidden semantic knowledge from a corpus of documents, reducing high-dimensional textual data into a manageable number of latent topics. Input texts are represented based on discovered latent topics. However, choosing the correct number of topics is difficult, as incorrect choices lead to suboptimal results or noisy topics. This type comprises 26.5% of

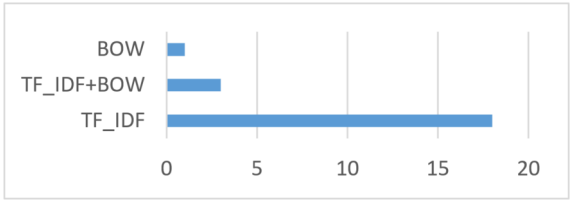

Fig.6: *The used VSM representations with their frequency.*

papers (13 out of 49). Six papers use LSA, one only uses LDA, and the remaining six use both LDA and LSA.

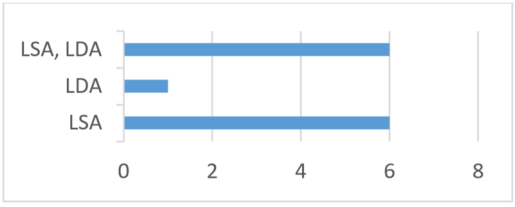

Fig.7: *The used Topic modelling techniques with their frequency.*

Advanced Embedding representation: In general, embedding techniques rely on the semantic representation of texts, and usually, embedding reduces the dimensions of the representation space compared to traditional methods. Embedding can exist at differ-

ent levels of detail, from word level to sentence level or even document level. This flexibility allows requirements to become represented at varying levels of abstraction. However, the embedding quality depends heavily on the training data, and some embedding generates vectors of fixed size, which can result in missing information about the varying importance or complexity of different aspects within the requirements. The representations proposed in the relevant papers include several techniques. This type comprises 46.9% of all documents (23 out of 49).

Word embedding: there are several techniques in this type (16 out of 49): word2vec, Bert, fast text, USE, Glove, etc. The following table shows the papers for each of them.

Table 2: *Word embedding techniques with their related papers.*

Word Embedding	Related Papers
Word2vec	[19] [26] [32] [39]
Bert	[37] [50] [52] [54] [59] [51]
Fast text	[46] [53]
USE	[60]
Glove	[55]
Embedding layer	[40] [58]

Statement embedding: documents that use this type of representation can be further classified based on the method used to merge word embedding to represent statements (5 out of 49). Table 3 shows all related papers classified into three types of statement embedding techniques (RNN, CNN, and ANN).

Table 3: *The used statement embedding techniques with their related papers.*

Statement Embedding	Related Papers
RNN based	[58] [32] [19]
CNN based	[58] [40]
ANN based	[53]

Document embedding: There are eight papers (16.3% of all papers) that use doc2vec for embedding input.

Natural language processing (NLP) approaches have seen significant developments. Where rule-based approaches and traditional information retrieval methods emerged, researchers then explored using ontologies and semantic networks to enhance language understanding. Later, the trend began towards statistical methods, leading to the methods of embedding and transformers.

According to the analysis of studies, a relative decline in traditional and simple methods was observed, with an increase in the use of embedding methods in the last two years.

RQ3: What are the proposed solutions in the literature for RT?

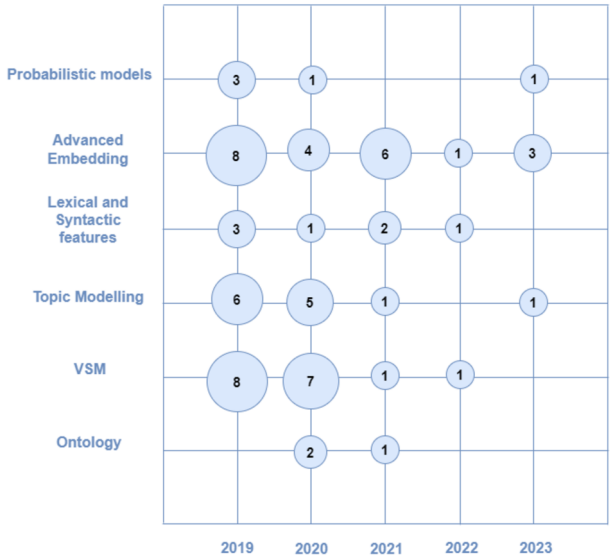

Fig.8: *Bubble chart showing the number of papers related to representation techniques and the year of publication.*

We covered two types of solving methods in this study, as shown in Fig. 9. 21 out of 49 papers use the rule-based approach (43% of all documents), and the remaining papers depend on the ML approach (57% of all documents). Fig. 10 shows the number of documents related to solving methods and the year of publication.

Fig.9: *Distribution of solving methods.*

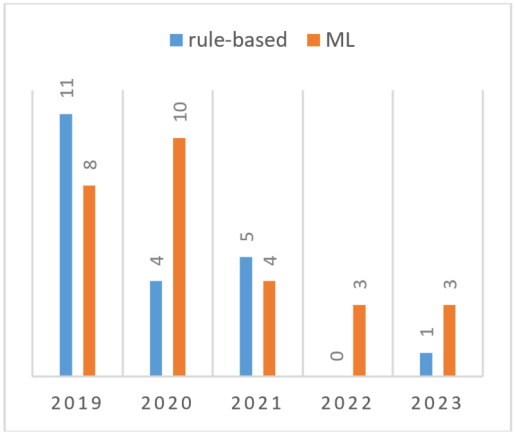

Fig.10: *Number of papers related to solving methods and the year of publication.*

28 out of 49 papers use the ML approach. In these, we found the most frequently used algorithms, as shown in Fig. 11. Other algorithms were also

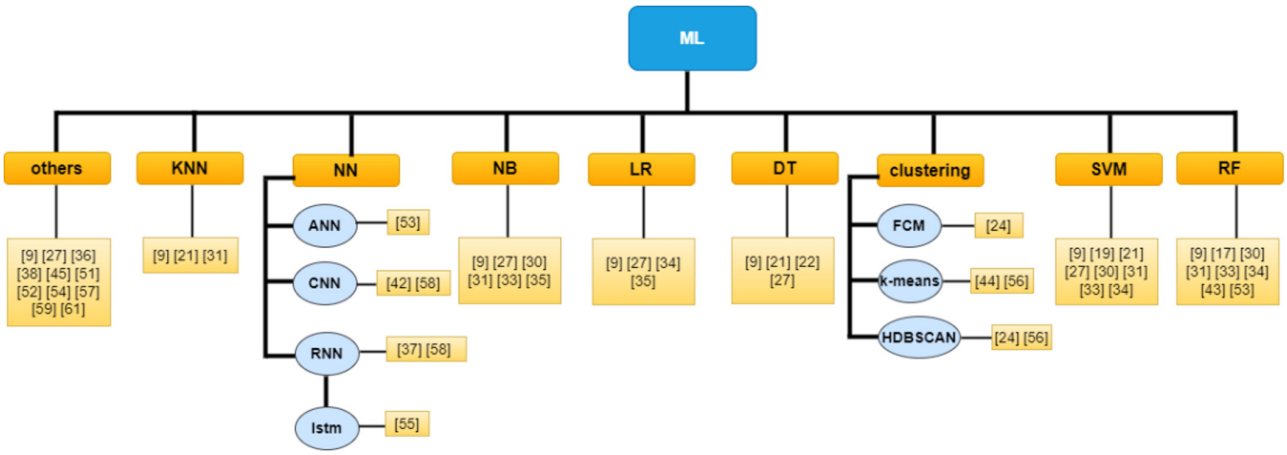

Fig.11: *ML algorithms with their related papers.*

used, such as GBDT [9], RBF and Part [27], GA [36], Bayesian Network [38], ABC [45], Association Rules [51], Greedy algorithm [57], FCA [61], and cost estimates [52] [54] [59].

Fig. 12 shows the number of papers related to ML techniques and the year of publication.

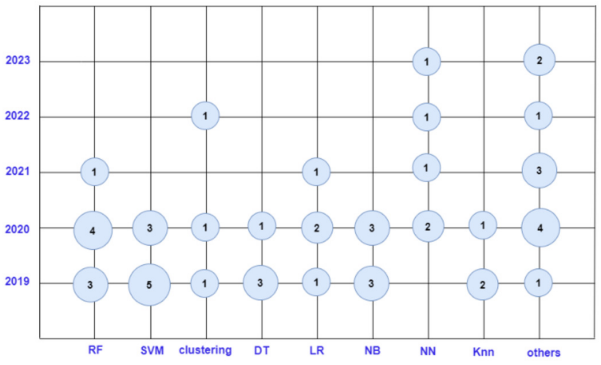

Fig.12: *The number of papers related to ML techniques and the year of publication.*

Q4: What are the types of RT in the literature?

In this study, we found two types of RT. The first concerns traceability between requirements only and is found in 36.7% of papers (18 out of 49). The other addresses traceability between other software artifacts and is found in 51% of papers (25 out of 49). The remaining 12.2% of papers (6 out of 49) studied the two types. Table 4 represents traceability details within related papers based on different types of artifacts. Fig. 13 displays the frequency of different artifacts used by researchers to demonstrate the traceability method.

RQ5: What are the datasets used in this field?

Our study showed the existence of many used datasets. We remarked that open-source datasets are more popular than closed ones. Fig. 14 shows the most frequently used datasets in the literature.

Table 4: *Types of traceability with related papers.*

Traceability artifacts	Related papers
Use Case to Code	[9] [14] [17] [18] [22] [25] [27] [34] [36] [38] [42] [55] [56]
Requirements to Code	[14] [18] [21] [23] [27] [28] [38] [42] [45] [46] [55] [58] [61]
Test Case to Code	[14] [15] [16] [17] [21] [27] [34] [49] [55]
Descriptions of Interaction Diagrams to Code	[14] [17] [25] [27] [55]
Descriptions of Interaction Diagrams to Test Case	[14] [17] [34] [55]
Use Case to Test Case	[14] [17] [25] [34] [36] [55]
Use Case-to-interaction Diagrams	[14] [17] [31] [55]
Requirements to Test Case	[36] [38]
Use case to UML (Class, Sequence) diagrams	[20]
High-level Requirement to Low-level Requirement	[17] [25] [34] [36] [55] [59]
Functional Requirements	[26] [29] [30] [61] [33] [35] [37] [39] [44] [47] [50] [51] [53] [54] [57] [60]
Non-Functional Requirements	[48]
Issue to Commit	[19] [32] [41] [43] [52]
Issue to Code, Test Case	[21] [40]
Test case	[24]

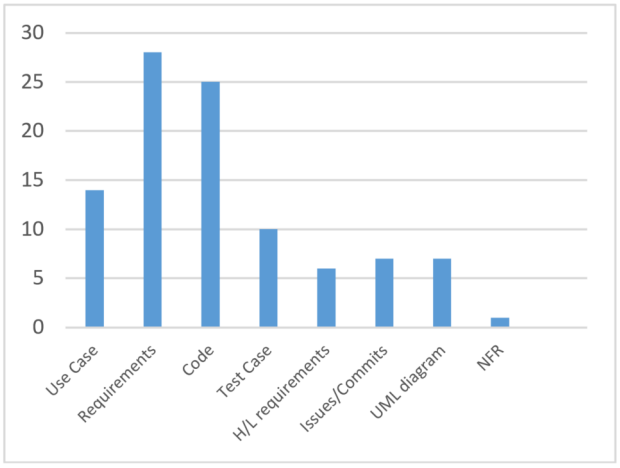

Fig.13: *Frequency of software artifacts used in the literature to demonstrate the traceability approach.*

These open datasets are commonly used in research to benchmark different approaches, providing developers with opportunities to contribute to or leverage them for evaluating and improving their RT processes. However, a notable limitation is that most studies using these datasets focus primarily on functional requirements, neglecting non-functional requirements (NFRs), which are crucial for defining system performance, scalability, and other critical constraints.

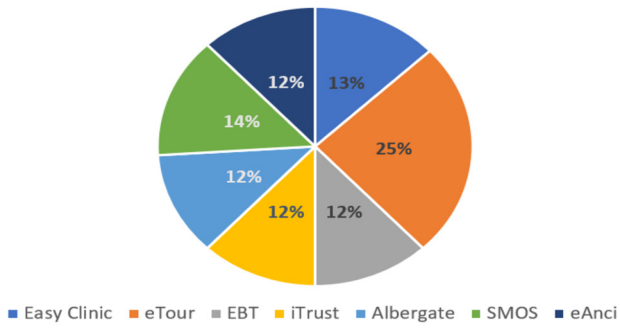

Fig.14: *The most frequent datasets used in literature.*

RQ6: What gaps and potential future directions exist in this field?

Based on our study, we observe that the used NLP representation impacts the performance of RT. Ontology-based representation performed well in horizontal and vertical traceability, whereas TF _IDF was the best in vertical traceability. The use of advanced embedding techniques to represent requirements seems to be promising for RT. Case studies from industries like healthcare or finance could demonstrate how embedding techniques improve traceability in compliance-critical projects, where accuracy is paramount.

On the other hand, this study shows that solutions rely on ML approaches and move more specifically towards DL techniques. Developers could benefit from adopting hybrid approaches that combine rule-based and ML methods to address specific challenges in traceability, such as handling ambiguous or incomplete requirements. For example, a telecom company could enhance traceability between high-level requirements and low-level implementation by integrating DL models with traditional methods.

We noticed that many use open-source datasets provided by CoEST, allowing for comparing different approaches. We also noticed that all studies that discovered traceability links between requirements and code relied only on functional requirements and did not consider non-functional requirements, constituting a limitation for these studies. Non-functional requirements impose numerous constraints that developers must implement in the code. Therefore, tracing them in the code before the software delivery is essential. For example, developers can extend embedding techniques to identify patterns related to non-functional requirements, while ontology-based approaches could incorporate predefined non-functional requirements taxonomies. Furthermore, creating standardized datasets with annotated non-functional requirements could enable machine learning models to enhance traceability accuracy. Standardized datasets annotated with NFRs could further help organisations and researchers strengthen the accuracy of machine-learning models for RT. A practical example includes using such datasets to improve the traceability of security requirements in e-commerce systems.

Another significant limitation is how word vectors are combined to formulate representations of requirements. Some research uses an RNN that focuses on predicting the next element in the sequence. This research could benefit from domain-specific representations or techniques that consider key syntactic roles in requirements (such as actor, action, objects,...) and semantic aspects of requirements, such as Req2Vec [39].

We noticed another limitation between the tracing process and its implementation when dealing with requirements with specific templates, especially in rule-based methods. Using "template-dependent" methods decreases accuracy when applied to new written requirements due to differences with the template or when using a completely different template, especially in large industrial and real-world projects.

We noticed a significant limitation concerning the lack of prioritization of requirements. Many studies have provided models to discover the relationships between requirements, such as dependency, causation, implicit relationships, etc. However, they did not handle the priority of requirements. For example, in safety-critical industries such as aerospace, prioritization ensures that essential safety features are traceable and validated before secondary requirements.

Case studies are in-depth, detailed explorations of specific examples or instances examining how theories, methodologies, or technologies apply in real-world scenarios, so including case studies in our paper can help:

• Ground theoretical discussions in practical, real-world examples.

• Showcase the effectiveness of the methods (like NLP in traceability).

• Provide detailed, quantified evidence to support claims and conclusions.

• Offer a clear comparison with existing studies or approaches, adding credibility and depth.

We had some case studies like the following:

Case Study 1: Vertical Requirements Traceability Using BERT [60]

Context: This study focused on improving vertical traceability between requirements and source code in a software development project. The aim was to enhance the accuracy of traceability links using BERT, a transformer-based model known for its contextual understanding of natural language.

NLP Technique Used: BERT (Bidirectional Encoder Representations from Transformers): The BERT model establishes links between software requirements and corresponding source code segments.

Implementation:

The BERT model uses a dataset of software requirements and annotated code files for training.

The model outputs a similarity score for each pair of requirements and code, indicating the likelihood of a traceability link.

Results: The case study shows a moderately positive correlation between requirements similarity and software similarity, with the pre-trained deep learning-based BERT language model with preprocessing outperforming the other models (Fast Text, Doc2vec, and TF_IDF).

Conclusion: The BERT model significantly improved traceability accuracy, demonstrating its effectiveness in handling complex software artifacts.

Case Study 2: Traceability Link Recovery Using Word2Vec and SVM [19]

Context: This study aimed to recover traceability links between issue reports and code commits in open-source projects. The objective was to enhance traceability using word embedding and a Support Vector Machine (SVM) classifier.

NLP Technique Used: Word2Vec: This model generated vector representations of words in issue reports and code commits, capturing semantic similarities.

SVM Classifier: The SVM classifier categorises whether a link exists between an issue report and a corresponding code commit.

Implementation: The dataset included issue reports and code commits from the Apache Software Foundation projects.

Word2Vec embedding converted textual data into numerical vectors, which we fed into the SVM classifier.

Results: On the ZOOKEEPER project, the model achieved a precision of 84.7%, a recall rate of 87%, and an F score of 85.9%.

Conclusion: Word2Vec combined with SVM proved to be a practical approach for recovering traceability links, offering a balance between precision and recall, which is suitable for large-scale projects.

Case Study 3: Topic Modelling for Requirements Categorisation [44]

Context: This case study classified requirements into functional and non-functional categories using Topic Modelling, particularly Latent Dirichlet Allocation (LDA) with K-means algorithm.

NLP Technique Used: LDA (Latent Dirichlet Allocation): We used LDA to discover latent topics in the requirements documents, which helped classify them into functional and non-functional categories.

Implementation:

The dataset consisted of 43 requirements from various documents.

LDA was applied to identify topics, which we then mapped to functional or non-functional requirements.

Results: The LDA model achieved a topic coherence score of 0.61, indicating a good match between the discovered topics and actual requirement categories.

Classification Accuracy: The average precision score from highest to lowest consecutively is in the IR + LDA (0.61 and 0.857).

Conclusion: The LDA effectively categorises requirements, particularly in distinguishing between functional and non-functional requirements. This categorisation is essential for ensuring that we have addressed all aspects of the software during development.

5. THREATS TO VALIDITY

We may not have been able to find the complete set of all relevant papers, but we have taken some measures to mitigate this threat.

• We chose six reputable and well-known data sources (Scopus, IEEE Xplore, ACM Digital Library, Elsevier, MDPI, and Springer Link) to maximize the number of candidate papers.

• Although we made the search list more general using several synonyms for each term, the final search list may not include all synonyms. We therefore checked the references of the final papers to add any additional relevant works.

• When there were doubts or conflicts about including an article, the authors discussed the final decision.

• We considered studies that used issue reports or bug reports. In other words, the studies are diverse,

and the primary focus of some might not be traceability. For instance, we included bug localization papers if the papers handled traceability of issue reports. From our point of view, such an inclusion is not a big deal because bug localization is one of the goals that traceability studies typically aim to achieve. Additionally, to mitigate this threat, we carefully determined whether the studies were related to traceability by checking which "trace" terms appeared in the studies.

6. CONCLUSION

This study has presented a systematic mapping review of relationships between software artifacts using NLP. Starting from 209 papers retrieved from six well-known digital libraries, we identified 49 primary documents that met the inclusion criteria. We analysed these works and classified them based on the targeted type of traceability, solution, and text representation technique.

Our findings reveal that approximately one-third of the reviewed studies address horizontal traceability, focusing on linking artifacts of similar types. At the same time, half explore vertical traceability, linking artifacts across different abstraction levels. Machine learning-based approaches dominate the proposed solutions, appearing in over half of the studies, with deep learning techniques gaining increasing attention. Advanced embedding methods, including word and contextualized embeddings, were employed in more than 45% of the reviewed studies, highlighting a trend toward more sophisticated requirement representation methods. Additionally, vector space models (VSM) are becoming more common for modelling requirements.

This review also uncovered key gaps and challenges in the field. Notably, non-functional requirements remain underexplored, and there is limited focus on adapting traceability techniques to agile workflows. These gaps point to significant opportunities for future research, such as developing methods to handle non-functional requirements effectively and creating traceability frameworks tailored to dynamic development environments.

The classification and insights derived from the selected studies provide a valuable reference for researchers and practitioners, offering a clear understanding of current trends and emerging directions in the field. By addressing the identified research gaps, future studies can further advance the use of NLP for requirements traceability, improving both the accuracy and applicability of these methods in software engineering practice.

AUTHOR CONTRIBUTIONS

Methodology, G.R. and N.G.; validation, G.R., N.G. and F.K.; formal analysis, F.K.; investigation, G.R., N.G.; writing—original draft preparation, F.K.; writing—review and editing, G.R., N.G., and F.K.; visualization, F.K.; supervision, G.R. and N.G. All authors have read and agreed to the published version of the manuscript.

References

[1] A. Tahir and R. Ahmad, "Requirement engineering practices-An empirical study," *Proceedings of the 2010 International Conference on Computational Intelligence and Software Engineering*, pp. 1-5, 2010.

[2] A. Aurum and C. Wohlin, "Requirements engineering: Setting the context," *Engineering and Managing Software Requirements*, pp. 1-15, 2005.

[3] X. Li, B. Wang, H. Wan, Y. Deng and Z. Wang, "Applications of Machine Learning in Requirements Traceability: A Systematic Mapping Study," in *Proceedings of the 35th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 566-571, 2023.

[4] J. Mucha, A. Kaufmann and D. Riehle , "A systematic literature review of pre-requirements specification traceability," *Requirements Engineering*, vol. 29, pp. 119-141, 2024.

[5] T. W. W. Aung, H. Huo and Y. Sui, "A literature review of automatic traceability links recovery for software change impact analysis," in *Proceedings of the 28th International Conference on Program Comprehension*, pp. 14-24, 2020.

[6] B. Wang, H. Wang, R. Luo, S. Zhang, Q. Zhu, "A Systematic Mapping Study of Information Retrieval Approaches Applied to Requirements Trace Recovery," in *Proceedings of the 34th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 1-6, 2022.

[7] Z. Pauzi and A. Capiluppi, "Applications of natural language processing in software traceability: A systematic mapping study," *Journal of Systems and Software*, vol. 198, p. 111616, 2023.

[8] Y. Lyu, H. Cho, P. Jung and S. Lee, "A Systematic Literature Review of Issue-Based Requirement Traceability," in *IEEE Access*, vol. 11, pp. 13334-13348, 2023

[9] T. Li, S. Wang, D. Lillis and Z. Yang, "Combining machine learning and logical reasoning to improve requirements traceability recovery," *Applied Sciences*, vol. 10, no. 20, p. 7253, 2020.

[10] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E. V. Chioasca and R. T. Batista-Navarro, "Natural language processing for requirements engineering: A systematic mapping study," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–41, 2021.

[11] O. C. Z. Gotel and C. W. Finkelstein, "An analysis of the requirements traceability problem," *Proceedings of IEEE International Conference*

on Requirements Engineering, Colorado Springs, CO, USA, pp. 94-101, 1994.

[12] S. Ibrahim, N. B. Idris, M. Munro, and A. Deraman, "A requirements traceability to support change impact analysis," *Asian Journal of Information Tech*, vol. 4, no. 4, pp. 345-355, 2005.

[13] R. Sonbol, G. Rebdawi and N. Ghneim, "The use of NLP-based text representation techniques to support requirement engineering tasks: A systematic mapping review," in *IEEE Access*, vol. 10, pp. 62811-62830, 2022.

[14] R. Tsuchiya, K. Nishikawa, H. Washizaki, Y. Fukazawa, Y. Shinohara, K. Oshima and R. Mibe, "Recovering transitive traceability links among various software artifacts for developers," *IEICE TRANSACTIONS on Information and Systems*, vol. 102, no. 9, pp. 1750–1760, 2019.

[15] V. Csuvik, A. Kicsi and L. Vidács, "Evaluation of textual similarity techniques in code-level traceability," *Computational Science and Its Applications—ICCSA 2019: 19th International Conference*, pp. 529–543, 2019.

[16] V. Csuvik, A. Kicsi and L. Vidács, "Source code level word embeddings in aiding semantic test-to-code traceability," in *2019 IEEE/ACM 10th International Symposium on Software and Systems Traceability (SST)*, pp. 29-36, 2019.

[17] C. Mills, J. Escobar-Avila, A. Bhattacharya, G. Kondyukov, S. Chakraborty and S. Haiduc, "Tracing with less data: Active learning for classification-based traceability link recovery," *Proceedings of the 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 103–113, 2019.

[18] H. Kuang, H. Gao, H. Hu, X. Ma, J. Lü, P. Mäder and A. Egyed, "Using frugal user feedback with closeness analysis on code to improve IR-based traceability recovery," *Proceedings of the 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, pp. 369–379, 2019.

[19] R. Xie, L. Chen, W. Ye, Z. Li, T. Hu, D. Du and S. Zhang, "DeepLink: A code knowledge graph-based deep learning approach for issue-commit link recovery," *Proceedings of the 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 434–444, 2019.

[20] D. Kchaou, N. Bouassida, M. Mefteh and H. Ben-Abdallah, "Recovering semantic traceability between requirements and design for change impact analysis," *Innovations in Systems and Software Engineering*, vol. 15, pp. 101-115, 2019.

[21] H. Abukwaik, A. Burger, B. K. Andam and T. Berger, "Semi-automated feature traceability with embedded annotations," *Proceedings of the 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 529–533, 2018.

[22] S. Wang, T. Li and Z. Yang, "Exploring semantics of software artifacts to improve requirements traceability recovery: A hybrid approach," *Proceedings of the 2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 39–46, 2019.

[23] D. V. Rodriguez and D. L. Carver, "Comparison of information retrieval techniques for traceability link recovery," in *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, pp. 186–193, 2019.

[24] S. Tahvili, L. Hatvani, M. Felderer, W. Afzal and M. Bohlin, "Automated functional dependency detection between test cases using doc2vec and clustering," in *2019 IEEE International Conference on Artificial Intelligence Testing (AITest)*, pp. 19–26, 2019.

[25] L. Chen, D. Wang, J. Wang and Q. Wang, "Enhancing unsupervised requirements traceability with sequential semantics," in *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 23–30, 2019.

[26] W. Alhoshan, R. Batista-Navarro and L. Zhao, "Using frame embeddings to identify semantically related software requirements," in *2nd Workshop on Natural Language Processing for Requirements Engineering*, 2019.

[27] A. H. Rasekh, S. M. Fakhrahmad and M. H. Sadreddini, "Mining traces between source code and textual documents," *International Journal of Computer Applications in Technology*, vol. 59, no. 1, pp. 43-52, 2019.

[28] N. Ali, H. Cai, A. Hamou-Lhadj and J. Hassine, "Exploiting parts-of-speech for effective automated requirements traceability," *Information and Software Technology*, vol. 106, pp. 126–141, 2019.

[29] M. Singh, "Using natural language processing and graph mining to explore inter-related requirements in software artefacts," *ACM SIGSOFT Software Engineering Notes*, vol. 44, no. 1, pp. 37–42, 2022.

[30] G. Deshpande, C. Arora and G. Ruhe, "Data-driven elicitation and optimization of dependencies between requirements," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pp. 416–421, 2019.

[31] R. Samer, M. Stettinger, M. Atas, A. Felfernig, G. Ruhe and G. Deshpande, "New approaches to the identification of dependencies between requirements," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1265–1270, 2019.

[32] H. Ruan, B. Chen, X. Peng and W. Zhao, "DeepLink: Recovering issue-commit links based on deep learning," *Journal of Systems and Software*, vol. 158, p. 110406, 2019.

[33] G. Deshpande, Q. Motger, C. Palomares, I. Kamra, K. Biesialska, X. Franch and J. Ho, "Requirements dependency extraction by integrating active learning with ontology-based retrieval," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pp. 78–89, 2020.

[34] T. B. Du, G. H. Shen, Z. Q. Huang, Y. S. Yu and D. X. Wu, "Automatic traceability link recovery via active learning," *Frontiers of Information Technology & Electronic Engineering*, vol. 21, no. 8, pp. 1217–1225, 2020.

[35] J. Frattini, M. Junker, M. Unterkalmsteiner and D. Mendez, "Automatic extraction of cause-effect relations from requirements artifacts," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pp. 561–572, 2020.

[36] B. Wang, R. Peng, Z. Wang, X. Wang and Y. Li, "An automated hybrid approach for generating requirements trace links," *International Journal of Software Engineering and Knowledge Engineering*, vol. 30, no. 07, pp. 1005–1048, 2020.

[37] J. Fischbach, B. Hauptmann, L. Konwitschny, D. Spies and A. Vogelsang, "Towards causality extraction from requirements," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pp. 388–393, 2020.

[38] K. Moran, D. N. Palacio, C. Bernal-Cárdenas, D. McCrystal, D. Poshyvanyk, C. Shenefiel and J. Johnson, "Improving the effectiveness of traceability link recovery using hierarchical Bayesian networks," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pp. 873–885, 2020.

[39] R. Sonbol, G. Rebdawi and N. Ghneim, "Towards a semantic representation for functional software requirements," in *2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pp. 1–8, 2020.

[40] L. R. J. Santos, G. Gadelha, F. Ramalho and T. Massoni, "Improving traceability recovery between bug reports and manual test cases," in *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, pp. 293–302, 2020.

[41] Y. Liu, J. Lin, and J. Cleland-Huang, "Traceability support for multi-lingual software projects," in *Proceedings of the 17th International Conference on Mining Software Repositories*, pp. 443–454, 2020.

[42] D. V. Rodriguez and D. L. Carver, "Multi-objective information retrieval-based NSGA-II optimization for requirements traceability recovery," in *2020 IEEE International Conference on Electro Information Technology (EIT)*, pp. 271–280, 2020.

[43] M. Rath, M. T. Tomova and P. Mäder, "Spojitr: Intelligently link development artifacts," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 652–656, 2020.

[44] R. Asyrofi, T. Hidayat and S. Rochimah, "Comparative studies of several methods for building simple traceability and identifying the quality aspects of requirements in SRS documents," in *2020 10th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, pp. 243-247, 2020.

[45] D. V. Rodriguez and D. L. Carver, "An IR-based artificial bee colony approach for traceability link recovery," in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1145–1153, 2020.

[46] T. Hey, F. Chen, S. Weigelt and W. F. Tichy, "Improving traceability link recovery using fine-grained requirements-to-code relations," in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 12–22, 2021.

[47] M. Aldekhail and M. Almasri, "Intelligent identification and resolution of software requirement conflicts: Assessment and evaluation," *Computer Systems Science & Engineering*, vol. 40, no. 2, 2022.

[48] U. Shah, S. Patel and D. C. Jinwala, "Detecting intra-conflicts in non-functional requirements," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 29, no. 03, pp. 435–461, 2021.

[49] A. Kicsi, V. Csuvik and L. Vidács, "Large scale evaluation of natural language processing-based test-to-code traceability approaches," in *IEEE Access*, vol. 9, pp. 79089–79104, 2021.

[50] S. Das, N. Deb, A. Cortesi and N. Chaki, "Sentence embedding models for similarity detection of software requirements," *SN Computer Science*, vol. 2, pp. 1–11, 2021.

[51] V. Leitão and I. Medeiros, "SRXCRM: Discovering association rules between system requirements and product specifications," in *REFSQ Workshops*, 2021.

[52] J. Lin, Y. Liu, Q. Zeng, M. Jiang and J. Cleland-Huang, "Traceability transformed: Generating more accurate links with pre-trained BERT models," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 324–335, 2021.

[53] A. Nicholson and G. J. LC, "Issue link label recovery and prediction for open source software," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pp. 126–135, 2021.

[54] G. Deshpande, B. Sheikhi, S. Chakka, D. L. Zotegouon, M. N. Masahati and G. Ruhe, "Is BERT the new silver bullet? - An empirical

investigation of requirements dependency classification," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pp. 36–145, 2021.

[55] J. Zhu, G. Xiao, Z. Zheng and Y. Sui, "Enhancing traceability link recovery with unlabeled data," in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, pp. 446–457, 2022.

[56] N. H. Al-walidi, S. S. Azab, A. Khamis and N. R. Darwish, "Clustering-based automated requirement trace retrieval," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 12, 2022.

[57] R. Sonbol, G. Rebdawi and N. Ghneim, "Learning software requirements syntax: An unsupervised approach to recognize templates," *Knowledge-Based Systems*, vol. 248, p. 108933, 2022.

[58] P. Dai, L. Yang, Y. Wang, D. Jin and Y. Gong, "Constructing traceability links between software requirements and source code based on neural networks," *Mathematics*, vol. 11, no. 2, p. 315, 2023.

[59] J. Tian, L. Zhang and X. Lian, "A cross-level requirement trace link update model based on bidirectional encoder representations from transformers," *Mathematics*, vol. 11, no. 3, p. 623, 2023.

[60] M. Abbas, A. Ferrari, A. Shatnawi, E. Enoiu, M. Saadatmand and D. Sundmark, "On the relationship between similar requirements and similar software: A case study in the railway domain," *Requirements Engineering*, vol. 28, no. 1, pp. 23–47, 2023.

[61] R. F. Al-Msie'deen, "Requirements traceability: Recovering and visualizing traceability links between requirements and source code of object-oriented software systems," *arXiv preprint arXiv:2307.05188*, 2023.

[62] L. C. Briand, Y. Labiche and L. O'Sullivan, "Impact analysis and change management of UML models," in *International Conference on Software Maintenance, 2003. ICSM 2003*, pp. 256–265, 2003.

[63] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf and S. Brinkkemper, "Improving agile requirements: The quality user story framework and tool," *Requirements Engineering*, vol. 21, pp. 383–403, 2016.

Fedaa Khalil received the B.S. degree in Computer Engineering from the Al-Wataniya Private University (WPU), Syria in 2021. She recently received her M.S. degrees in Information Engineering at the Higher Institute for Applied Sciences and Technology (HIAST), Syria in 2025. Passionate about advancing technology, she is now pursuing a PhD in Information Engineering, further deepening her expertise in the field. Throughout her academic journey, she has contributed to several research papers and has actively participated in applied projects, particularly in the realm of artificial intelligence. Her research interests include natural language processing and software engineering.

Ghaida Rebdawi received the B.S. degree in informatics engineering from the Higher Institute for Applied Sciences and Technology, Syria, in 1985, and the DEA and Ph.D. degrees in applied automatic and informatics from the Institut National des Sciences Appliquées de Lyon, France, in 1987 and 1990, respectively. In 1991, she joined as a Research Assistant Professor at the Informatics Department, Higher Institute for Applied Sciences and Technology, where she has been a Research director/ Professor, since 2012. She coauthored many academic books in computer science, and more than 20 articles. Her research interests include software engineering, requirements engineering, natural language processing, and business process management.

Nada Ghneim received the Ph.D. degree (DEA) in artificial intelligence (image, robotics, vision) from the National High School of Computer Science and Applied Mathematics in Grenoble (ENSIMAG), France, in 1993, and the Ph.D. degree in language sciences (speech communication) from the Institut de la Communication Parlée, Stendhal (Grenoble III) University, France, in 1997. She is an Assistant Professor with the Faculty of Informatics Communication Engineering, Arab International University (AIU), Damascus, Syria. She is also a Researcher/Lecturer with the Higher Institute for Applied Sciences and Technology (HIAST) and the Information Technology Engineering Faculty, Damascus University. She has many publications in her research areas, including AI, speech and natural language processing.