IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 31, No. 4, April, 2026, pp. 1307-1322
Journal homepage: http://iieta.org/journals/isi

Traceable Reverse Engineering of UML Artifacts from SRS/SDD Documents Using NLP-Based Text Processing and Siamese BiLSTM: An IdVar4CL Case Study

Yudi Priyadi[1,2*], Eko Darwiyanto[3,4], Rio Nurtantyana[1,4,5]

[1] Center of Excellence of Artificial Intelligence for Learning and Optimization, Telkom University, Bandung 40257, Indonesia
[2] Software Engineering, Telkom University, Bandung 40257, Indonesia
[3] Center of Excellence HUMIC, School of Computing, Telkom University, Bandung 40257, Indonesia
[4] Informatics, Telkom University, Bandung 40257, Indonesia
[5] Research Center for Data and Information Sciences, National Research and Innovation Agency, Bandung 40135, Indonesia

Corresponding Author Email: whyphi@telkomuniversity.ac.id

https://doi.org/10.18280/isi.310425

Received: 8 February 2026
Revised: 5 April 2026
Accepted: 20 April 2026
Available online: 30 April 2026

Keywords:
Unified Modeling Language reverse engineering, requirements traceability, Software Requirements Specification, Siamese Bidirectional Long Short-Term Memor, software artifact alignment, Unified Specification Document, IdVar4CL

ABSTRACT

Reverse engineering Unified Modeling Language (UML) artifacts from Software Requirements Specification (SRS) and Software Design Description (SDD) documents remains difficult because requirement statements, use cases, scenario steps, and design representations are often created separately and drift semantically over time. This study develops a traceability-oriented pipeline for the IdVar4CL Unified Specification Document (USD) that combines rule-based text preprocessing with Siamese Bidirectional Long Short-Term Memory (BiLSTM) pair classification. The pipeline performs tokenization, normalization, stopword removal, stemming, semantic pair construction, and binary classification across Functional Requirement–Use Case (FR–UC), Use Case–Step Performed (UC–SP), and Functional Requirement–Step Performed (FR–SP) artifact pairs. A case dataset consisting of 15 labeled IdVar4CL documents was expanded into 75 inter-document pairs through a one-to-many relationship scheme. Stratified five-fold validation yielded an average accuracy of 0.9733 ± 0.0435, precision of 0.9542 ± 0.0728, recall of 1.0000 ± 0.0000, F1-score of 0.9754 ± 0.0396, and ROC-AUC of 1.0000 ± 0.0000. The predicted links were then transformed into explicit FR→UC→SP paths, producing an updated USD with end-to-end traceability between requirements and behavioral descriptions. The case study shows that Siamese BiLSTM can support consistent artifact linkage in small but structured specification datasets. Remaining false positives indicate that stronger negative-pair sampling, semantic validation, and human review are still needed before wider deployment.

1. INTRODUCTION

Software Engineering (SE) is increasingly shifting toward Artificial Intelligence (AI)-driven engineering, emphasizing process automation, quality improvement, and faster development cycles across industry and academia [1]. In Software Requirements Engineering (SRE), advances include the use of Design Thinking for richer elicitation [2], Recommender Systems for predicting or suggesting requirements [3], and systematic reviews on Unified Modeling Language (UML) diagram utilization in SE research [4-9]. In parallel, Natural Language Processing (NLP)—including terminology extraction and ontology construction [10], Exabsum-based summarization for information compression [11], and data augmentation for improving model generalization [12, 13]—has enhanced the accuracy of unstructured text analysis. Nevertheless, integrating cross-text artifact semantics between Software Requirements Specification (SRS) and Software Design Description (SDD) to ensure consistent design representation across Use Case, Sequence, Class, and State Diagrams remains an open and significant research challenge.

Several gaps and challenges remain. First, many AI studies in SE emphasize conceptual frameworks or automation at specific stages (e.g., testing, debugging, predictive maintenance) but lack deep text processing to preserve semantic coherence between documentation artifacts and diagrams [1, 4]. Second, studies on Identification Variable for Causal Loop (IdVar4CL) have demonstrated the extraction of causal variables from system narratives via text mining [8] and shown potential for semantic integration in Use Case Scenarios (UCSs) and FR/NFR artifacts [6, 7]; however, these processes remain manual and have not yet leveraged bidirectional sequential models such as Bidirectional Long Short-Term Memory (BiLSTM) to enhance consistency and accuracy. Third, while BiLSTM has proven effective at capturing bidirectional context in text-based tasks—such as fake review detection using BiLSTM with attention and positional embeddings [14] and code evaluation/improvement—its application to the reverse

engineering of SRS/SDD-based UML artifacts remains limited. Furthermore, ensuring trustworthiness and explainability requires a robust pipeline that maintains data quality and transparent process traceability to produce reliable reverse-engineering outcomes [15-17]. Overall, this highlights a critical research gap in the lack of an integrated approach combining advanced text processing and BiLSTM for semantically consistent reverse engineering of UML artifacts from SRS/SDD documents.

Our research is motivated by the practical need in the field that development teams require a measurable and reproducible way to (i) extract information units from SRS/SDD, (ii) map them to UML artifacts consistently (e.g., Use Case/Sequence Diagram), and (iii) translate the results into a Unified Specification Document (USD) that minimizes semantic drift. The IdVar4CL artifact prototype, which produces mappings between Architecturally Significant Requirements (ASR) and Quality Attributes (QA) [5], UCSs Alignment [6] and FR/NFR formation [7], has shown the right direction, but requires improvement through a two-way sequential model and Machine Learning-based validation flow to ensure robustness and traceability in line with the Responsible AI / Trustworthy AI agenda [15-17]. Thus, adopting BiLSTM, which has been proven effective for text/sequence modeling [14, 18-21], is considered appropriate to overcome the limitations of manual approaches and to build an accurate and transparent reverse-engineering pipeline.

In line with the broader roadmap of our previous research [5-7], the main objective of this study is to propose and evaluate a text-processing and Siamese BiLSTM-based reverse engineering pipeline for UML-related artifacts in the IdVar4CL case. The pipeline is designed not only to infer semantic relationships among documentation artifacts but also to generate a complete and traceable USD by constructing explicit FR→UC→SP paths. Specifically, this study performs the following activities:

(1) Capturing the bidirectional context of SRS/SDD through BiLSTM.
(2) Generating semantically consistent UML artifacts, such as FR/NFR, UCDs, and UCSs.
(3) Compiling an updated IdVar4CL USD that explicitly represents traceable FR→UC, UC→SP, FR→SP, and end-to-end FR→UC→SP paths as an integrated specification-design artifact.
(4) Ensuring trustworthiness through data validation and measurable evaluation (accuracy/consistency).

The main contributions of this research are as follows:

(1) An integrated reverse-engineering pipeline that converts written requirements into UML-related artifacts and USD. This study presents a pipeline that uses text preprocessing and Siamese BiLSTM-based pair classification. It identifies semantic relationships among Functional Requirements, Use Cases, and Steps Performed.
(2) Generation of traceable paths linking Functional Requirements (FR) to Use Cases (UC) and Steps Performed (SP). The approach predicts whether document pairs are linked or unlinked. It then transforms these results into clear traceability paths that connect Functional Requirements, Use Cases, and Steps Performed. These paths form the structural basis of the updated IdVar4CL USD.
(3) A complete and traceable IdVar4CL USD. The updated USD preserves the artifact layers: FR, UC, and SP. It shows their directional relationships using FR→UC, UC→SP, and FR→SP mappings. This supports specification-design consistency and reduces semantic drift.
(4) Evaluation and validation of artifact linkage inference. The pipeline is evaluated using stratified k-fold validation. Standard classification metrics, including accuracy, precision, recall, F1-score, and ROC-AUC, measure the reliability of semantic linkage inference.

In terms of scientific implications and impacts, this research addresses the integration gap between NLP and BiLSTM for Reverse Engineering UML from unstructured specification documents. Practically, the proposed pipeline provides a reproducible path for SE teams to align SRS/SDD, UML, and USD with a clear process trail (in line with Responsible/Trustworthy AI good practices) [15-17]. This approach can also be extended to other domains with semantic integration challenges (e.g., Knowledge Graphs/Ontologies [18]) and to large-scale data scenarios (e.g., big data summarization [22-26]).

Furthermore, the organization of this paper includes the following sections: Section 2 reviews related work (SRE, NLP, BiLSTM, Trustworthy/Responsible AI) that underlies this research [1-27]; Section 3 defines the use of datasets derived from UML artifacts called IdVar4CL. Section 4 describes the methods and pipeline (text preprocessing, BiLSTM process, mapping to UML, USD compilation, and evaluation scheme); Section 5 presents the experimental results and analysis; Section 6 summarizes the results, contributions, and future work plans.

2. RELATED WORKS

This section systematically reviews previous studies related to AI and NLP in software engineering documentation, semantic extraction from requirements and design documents, BiLSTM-based sequential semantic modeling, and traceability-oriented reverse engineering of UML artifacts. The review is organized to clarify the contribution and limitations of previous studies in supporting semantic extraction, cross-artifact alignment, and traceability construction across FR, UC, and SP. Furthermore, this section identifies the research gap addressed by the proposed Siamese BiLSTM-based reverse-engineering pipeline for the IdVar4CL USD.

2.1 Artificial Intelligence and Natural Language Processing in software engineering documentation

AI has increasingly influenced software engineering by enabling automation, improving quality, and accelerating development throughout the software development lifecycle. Alenezi and Akour [1] discuss AI-driven innovations in software engineering, including automated testing, intelligent debugging, and predictive maintenance. Their work demonstrates the broad adoption of AI in software engineering processes; however, it remains focused on general lifecycle automation and does not specifically address semantic extraction from SRS and SDD documents or traceability among UML-related artifacts.

In SRE, several studies have explored techniques for improving requirements elicitation and documentation quality.

Martins et al. [2] demonstrate the use of Design Thinking to support requirements elicitation through human-centered interaction. Priyadi [3] proposes a method for Software Requirement Specification compatibility design using text mining of UCD artifacts. Koç et al. [4] provide a systematic literature review of UML diagrams in software engineering research and show that UML remains important for software modeling and communication. Nevertheless, these studies do not explicitly address semantic linkage inference among requirement statements, use cases, and scenario-level behavioral descriptions.

NLP has also become an important foundation for analyzing unstructured software documentation. Xu et al. [10] review terminology extraction methods from textual corpora, emphasizing semantic representation and information retrieval. Merrouni et al. [11] propose EXABSUM, a hybrid extractive–abstractive text summarization approach. Pais et al. [12] discuss NLP-as-a-Service platforms for scalable NLP orchestration, while Shorten et al. [13] review text data augmentation techniques for improving deep learning generalization. These studies demonstrate that NLP can support terminology extraction, summarization, representation learning, and text enhancement; however, their objectives are generally not focused on UML artifact reverse engineering or on traceability-oriented USD construction from SRS/SDD artifacts.

Therefore, although AI and NLP studies provide strong foundations for software documentation analysis, previous works remain limited in connecting textual requirements, use cases, and scenario-level artifacts into explicit traceability structures such as FR→UC→SP paths.

2.2 Semantic extraction from requirements and design documents

Semantic extraction from requirements and design documents is essential because software artifacts are commonly represented in natural language and contain implicit relationships among actors, actions, objects, constraints, and process flows. In the context of IdVar4CL, several previous studies have explored semantic processing for artifact construction and transformation. Seba et al. [5] apply text processing to map Architecturally Significant Requirements (ASR) to Quality Attributes (QA). Laksana et al. [6] generate UCSs based on UCDs using text semantics, while Fatimatuzzahra et al. [7] identify FR and Non-Functional Requirements (NFR) from requirement elicitation artifacts using semantic analysis. These studies demonstrate that semantic extraction can support the formation and alignment of software artifacts.

Other IdVar4CL-related studies also demonstrate the usefulness of semantic extraction in software engineering artifacts. Priyadi et al. [8] introduce IdVar4CL, a causal loop variable identification method based on text mining for systems thinking analysis. Rachmanda et al. [9] identify Sequence Diagram objects from semantically processed UCSs. These studies confirm that semantic text processing can transform textual narratives into software engineering artifacts and behavioral representations.

Despite these contributions, previous semantic extraction studies remain limited to partial artifact processing or one-directional transformations. Existing works mainly focus on ASR ⇄ QA mapping, FR/NFR identification, scenario generation, causal loop variable extraction, or sequence object identification. They do not yet provide an integrated mechanism for preserving semantic relationships among FR, UC, and SP artifacts within a single traceability-oriented USD structure. Furthermore, previous studies do not explicitly formulate semantic alignment as a pair-classification problem between heterogeneous software artifacts.

In reverse engineering of UML-related artifacts, semantic extraction should not only identify individual artifacts but also preserve directional relationships among them. Functional Requirements should be connected to one or more Use Cases, and each Use Case should be linked to corresponding scenario-level Steps Performed. Therefore, the present study extends previous IdVar4CL semantic-processing research by inferring FR–UC, UC–SP, and FR–SP relationships and transforming them into traceable FR→UC→SP paths in the updated USD.

2.3 Bidirectional Long Short-Term Memory for sequential semantic modeling

BiLSTM is relevant for semantic extraction because it processes textual sequences in both forward and backward directions. This capability allows the model to capture contextual dependencies before and after a token, which is important for software documentation because requirement and scenario statements often depend on the interaction among actors, modal verbs, actions, objects, and process sequences. For example, statements such as “Researcher must upload storytelling documents” and “IdVar4CL Application must receive the document” contain semantic relationships that cannot be fully represented through isolated keyword matching alone.

Several previous studies demonstrate the effectiveness of BiLSTM and sequential models in text and software-related domains. Chen et al. [14] apply an attention-based BiLSTM with positional embeddings for fake review detection, demonstrating that BiLSTMs improve contextual semantic representations in classification tasks. Rahman et al. [21] apply a bidirectional LSTM language model for code evaluation and repair, demonstrating the BiLSTM's ability to capture sequential structures in software artifacts. Bai et al. [22] combines Temporal Convolutional Networks (TCN) with BiLSTM for temporal fault prediction in industrial equipment, further confirming the ability of sequential models to learn temporal contextual dependencies. Du et al. [23] review fault detection and state estimation techniques in automatic control systems, emphasizing the importance of sequential dependency modeling in complex systems.

Sequential and semantic modeling have also been explored in adjacent domains. Kutay and Yener [24] investigate semantic communication using pretrained models to efficiently represent text and images. Patil and Gudivada [25] reviews current trends and challenges in Large Language Models (LLMs), including contextual representation learning, scalability, and semantic encoding. Nagwani [26] applies topic modeling and clustering for summarizing large-scale text collections in big data environments. These studies contribute to contextual learning, semantic encoding, and large-scale text processing; however, they are not specifically directed toward UML artifact reverse engineering or traceability-oriented USD construction from SRS/SDD documents.

Although previous studies demonstrate the effectiveness of BiLSTM and sequential semantic modeling, their objectives are generally focused on classification, detection, summarization, communication efficiency, temporal

prediction, or code repair. They do not explicitly address cross-artifact semantic alignment among Functional Requirements, Use Cases, and Steps Performed. Furthermore, prior work does not transform semantic predictions into traceability-oriented software documentation artifacts, such as the USD.

In contrast, the present study applies a Siamese BiLSTM architecture for artifact-pair classification. Two artifact texts are encoded using a shared BiLSTM encoder, and their representations are compared to infer whether the pair is semantically linked or unlinked. This design is suitable for FR–UC, UC–SP, and FR–SP relationship inference because it allows the model to learn semantic compatibility between heterogeneous software artifacts. More importantly, the predicted relationships are transformed into explicit FR→UC→SP traceability paths in the updated IdVar4CL USD. Therefore, this study extends the role of BiLSTM from general semantic classification to traceability-oriented reverse engineering of UML artifacts.

2.4 Traceability and reverse engineering of Unified Modeling Language artifacts

Traceability is a critical requirement in reverse engineering because software documentation should preserve relationships between specification-level and design-level artifacts. In this study, traceability refers to the ability to connect Functional Requirements to Use Cases and Steps Performed through explicit FR→UC→SP paths. Without such traceability, requirements may become disconnected from their behavioral realization, leading to semantic drift between SRS/SDD documents and UML artifacts.

Previous studies have discussed UML diagrams, semantic representations, and software documentation from different perspectives. Koç et al. [4] review UML diagrams in software engineering research but focus mainly on UML classification and trends rather than semantic reverse engineering. Yahya et al. [18] discuss Semantic Web and Knowledge Graphs for Industry 4.0 interoperability, demonstrating the usefulness of semantic representation for integrating heterogeneous data. Li et al. [19] combine semantic and structural graph information for binary code similarity detection, while Cohen et al. [20] use graph-based semantic analysis to identify obfuscated binary code. These studies highlight the importance of semantic and structural relationships; however, their focus is on industrial interoperability or low-level binary code representation rather than on the construction of textual UML artifacts from SRS/SDD documentation.

Trustworthiness and explainability are also important for AI-supported software engineering and reverse engineering pipelines. Zhang et al. [15] propose a hierarchical evaluation framework for data quality in trustworthy AI. Shamsuddin et al. [16] discuss the integration of responsible AI and explainability, while Adadi and Berrada [17] surveys Explainable AI (XAI) techniques. Kowald et al. [27] discuss challenges in establishing and evaluating trustworthy AI systems. These studies emphasize transparency, accountability, and data quality in AI systems; however, they do not explicitly address how semantic predictions from SRS/SDD artifacts can be transformed into auditable traceability matrices or FR→UC→SP paths.

The present study positions traceability as the final output of the reverse-engineering pipeline. The proposed approach does not stop at predicting whether two software artifacts are semantically related. Instead, the predicted FR–UC, UC–SP, and FR–SP relationships are transformed into a USD structure that supports forward and backward traceability. Forward traceability enables stakeholders to inspect how requirements are realized through use cases and scenario steps, while backward traceability enables scenario-level behavior to be traced back to the originating requirements. Consequently, the generated USD supports requirements validation, design consistency checking, and UML reverse engineering with traceability.

2.5 Comparative analysis and research gap

To clarify the novelty of the proposed approach, Table 1 compares representative previous studies with the present study based on their focus, methods, artifact scope, limitations, and differences. The comparison shows that previous works have contributed to AI-supported software engineering, semantic extraction, BiLSTM-based modeling, trustworthy AI, and software artifact processing. However, most previous studies remain limited to conceptual review, single-artifact processing, classification, or partial transformation among software artifacts. In contrast, this study integrates text preprocessing, Siamese BiLSTM-based pair classification, and traceability-oriented USD construction to generate explicit FR→UC→SP paths.

Based on the reviewed studies, four main findings can be synthesized. First, AI in software engineering has largely focused on lifecycle automation, conceptual frameworks, testing, debugging, and predictive maintenance rather than semantic integration of UML-related artifacts. Second, NLP-based studies demonstrate strong capabilities for semantic extraction, summarization, and contextual representation but generally do not construct traceable links among software artifacts. Third, BiLSTM and sequential models have been effective in classification, code evaluation, communication, and temporal prediction tasks, yet their application to UML artifact reverse engineering remains limited. Fourth, trustworthy and explainable AI studies emphasize transparency and accountability but do not directly address traceability-oriented USD construction.

Table 1. Comparative analysis and research gap

Research Stream	Main Focus	Method/ Technique	Artifact Scope	Limitation	Difference and Advantage of This Study
Alenezi and Akour [1]	AI-driven innovation in SE	AI review	General SE lifecycle	No UML traceability	Focuses on traceable UML reverse engineering
Martins et al. [2]	Requirements elicitation	Design Thinking	Requirement elicitation	Manual interaction	Uses automated semantic linkage inference
Priyadi [3]	Requirement compatibility	Text mining	SRS and UC artifacts	No BiLSTM pair modeling	Generates FR→UC→SP traceability
Koç et al. [4]	UML studies	Literature	UML diagrams	No reverse	Provides operational UML reverse

		review		engine pipeline	engineering
Seba et al. [5]	ASR–QA map	Text processing	ASR and QA	Limited artifact	Extends to FR–UC–SP traceability
Laksana et al. [6]	UCS generation	Text semantics	UCD and UCS	No traceability modeling	Generates explicit FR→UC→SP paths
Fatimatuzzahra et al. [7]	FR/NFR formation	Semantic extraction	Requirement artifacts	No cross-artifact inference	Supports USD traceability construction
Priyadi et al. [8]	CLD variable extraction	Text mining	Narrative text	No UML reverse engine	Uses IdVar4CL for UML-related traceability
Rachmanda et al. [9]	Sequence object identification	Text semantics	UCS and Seq. Diagram	Partial artifact transformation	Provides integrated FR–UC–SP linkage
Xu et al. [10]	Terminology extraction	NLP extraction	Text corpora	No UML traceability	Uses NLP for semantic linkage inference
Merrouni et al. [11]	Text summarization	Hybrid summarization	Text documents	No artifact linkage	Preserves software artifact relationships
Pais et al. [12]	NLP services	NLP platform	NLP workflow	No UML reverse engine	Applies NLP in UML traceability pipeline
Shorten et al. [13]	Text augmentation	Deep learning augmentation	NLP datasets	No traceability focus	Focuses on traceable software artifacts
Chen et al. [14]	Fake review detection	Attention-based BiLSTM	Review text	No UML reverse engine	Uses Siamese BiLSTM for FR–UC–SP inference
Zhang et al. [15], Shamsuddin et al. [16], Adadi and Berrada [17], Kowald et al. [27]	Trustworthy and explainable AI	Transparency and explainability frameworks	General AI systems	No software traceability mechanism	Incorporates traceability-oriented USD validation
Yahya et al. [18]	Semantic Web and KG	Semantic interoperability	Industry 4.0 data	No SRS/SDD reverse engine	Applies semantic alignment to software artifacts
Li et al. [19], Cohen et al. [20]	Semantic graph analysis	Graph-based semantic	Binary code	Low-level representation	Focuses on textual UML-related artifacts
Rahman et al. [21]	Code evaluation and repair	BiLSTM	Source code	No artifact traceability	Applies Siamese BiLSTM to software documentation
Bai et al. [22], Du et al. [23]	Sequential temporal modeling	TCN–BiLSTM and control analysis	Temporal systems	No UML reverse engineering	Uses sequential modeling for semantic linkage
Kutay and Yener [24]	Semantic communication	Pretrained semantic encoding	Text and image communication	No software documentation traceability	Focuses on FR–UC–SP semantic alignment
Patil and Gudivada [25]	LLM review	LLM analysis	LLM	No UML artifact reverse Eng.	Uses lightweight Siamese BiLSTM for traceability
Nagwani [26]	Big data summarization	Topic modeling and clustering	Large text collections	No cross-artifact traceability	Focuses on USD-oriented semantic linkage
Proposed Study	Traceable UML artifact reverse engineering	Text preprocessing and BiLSTM	FR, UC, and SP artifacts	—	Generates complete and traceable FR→UC→SP paths in the updated USD

Note: SE: Software Engineering; SRS: Software Requirements Specification; SDD: Software Design Description; NLP: Natural Language Processing; LLM: Large Language Model; UML: Unified Modeling Language; BiLSTM: Bidirectional Long Short-Term Memory; FR: Functional Requirement; UC: Use Case; SP: Step Performed; USD: Unified Specification Document; UCS: Use Case Scenario.

These limitations reveal a research gap in the absence of an integrated approach that combines text preprocessing, Siamese BiLSTM-based semantic pair modeling, and traceability-oriented USD construction. Existing studies have not sufficiently addressed how Functional Requirements, Use Cases, and Steps Performed can be semantically linked and organized into explicit FR→UC→SP paths. Therefore, this study proposes a Siamese BiLSTM-based reverse-engineering pipeline that infers FR–UC, UC–SP, and FR–SP relationships and transforms them into a complete and traceable IdVar4CL USD.

3. DATASETS

The research activity focuses on the Reverse Engineering of UML artifacts using datasets derived from the SRS/SDD document named IdVar4CL. Not all IdVar4CL artifacts are used in this paper; instead, their selection aligns with the research's scope and objectives on Reverse Engineering. Therefore, three artifacts are selected as datasets: FR, UCDs, and UCSs.

Table 2. Functional Requirement (FR)

ID	Functional Requirement	Use Case Name
FR-CL01	Researcher must upload storytelling documents into the application interface.	Upload Storytelling Document
FR-CL02	Researcher must initiate text preprocessing on uploaded documents including case folding, tokenization, stopword removal, and stemming.	Initiate Text Preprocessing
FR-CL03	Researcher must calculate similarity between document pairs using TF-IDF and semantic similarity algorithms.	Calculate Similarity Documents
FR-CL04	Researcher must extract candidate variables from word pairs with semantic similarity scores above 0.50.	Extract Candidate Variables
FR-CL05	Systems Analyst must construct causal loop diagrams using the extracted variables.	Construct Causal Loop Diagram

Following the requirements elicitation activities that had been conducted in previous studies [5-9], FR are defined as statements that follow the sentence pattern Subject + Modals + Verb + Object. The FR documents used as reference sources for deriving subsequent artifacts are presented in Table 2.

Still referring to Table 2, five documents are used as reference datasets, as follows:

(1) The researcher must upload storytelling documents into the application interface.
(2) The researcher must initiate text preprocessing on uploaded documents, including case folding, tokenization, stopword removal, and stemming.
(3) The researcher must calculate similarity between document pairs using TF-IDF and semantic similarity algorithms.
(4) The researcher must extract candidate variables from word pairs with semantic similarity scores above 0.50.
(5) Systems Analyst must construct causal loop diagrams using the extracted variables.

The next artifact is the UCD. This artifact consists of five UC. Ideally, the number of Use Cases should be equal to or greater than the number of Functional Requirements. Referring to Figure 1, two Actors and five Use Cases are defined, all of which are used as datasets. The resulting requirement statements derived from the UCD are as follows:
(1) Actors consisting of a Researcher and a Systems Analyst.
(2) Use Cases consisting of Upload Storytelling Document, Initiate Text Preprocessing, Calculate Similarity Documents, Extract Candidate Variables, and Construct Causal Loop Diagram.

The third artifact used in this study is the UCS. In this artifact, the requirement statements used are based on the Step Performed (SP) section, where all statements for all steps use the sentence pattern Subject + Modals + Verb + Object. Please see Figure 2.

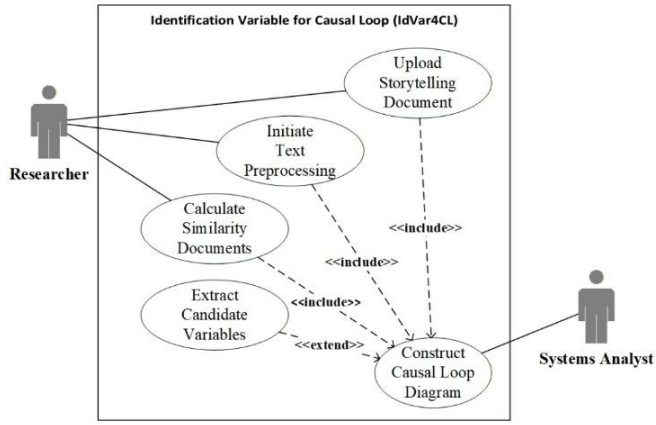

Figure 1. Use Case Diagram (UCD)

Usecase Name	Calculate Similarity Documents	ID : SP-CL03
Actor	Researcher	
Description	Researcher calculate similarity between document pairs using TF-IDF and semantic similarity.	
Step Performed	1. Researcher should select document pairs for similarity calculation. 2. IdVar4CL Application must verify selected documents are available. 3. Researcher should initiate similarity calculation. 4. IdVar4CL Application must calculate similarity using TF-IDF and semantic algorithms.	

Usecase Name	Initiate Text Preprocessing	ID : SP-CL02
Actor	Researcher	
Description	Researcher preprocess the documents to prepare for analysis	
	1. Researcher should access the preprocessing section. 2. IdVar4CL Application must show the preprocessing interf 3. Researcher must select uploaded documents. 4. IdVar4CL Application should verify document availability	

Usecase Name	Extract Candidate Variables	ID : SP-CL04
Actor	Researcher	
Description	Researcher extract candidate variables from word pairs with semantic similarity scores above 0.50.	
	1. Researcher should select word pairs with similarity scores above 0.50.	

Usecase Name	Upload Storytelling Document	ID : SP-CL01
Actor	Researcher	
Description	Researcher input documents into the IdVar4CL Application.	
Step Performed	1. Researcher should access the data input form. 2. IdVar4CL Application must display the page. 3. Researcher should upload the document. 4. IdVar4CL Application must receive the document from the researcher.	
Preconditions	Document input form is shown	
Postconditions	IdVar4CL Application should successfully receive the document.	
Assumptions	Researcher uploads a supported document type within the allowed size lin	

Usecase Name	Construct Causal Loop Diagram	ID : SP-CL05
Actor	Systems Analyst	
Description	Systems Analyst build causal loop diagrams using extracted variables.	
Step Performed	1. Systems Analyst should select extracted variables. 2. IdVar4CL Application must verify the variables are available. 3. Systems Analyst should initiate diagram construction. 4. IdVar4CL Application must generate causal loop diagrams from the variables. 5. IdVar4CL Application should display the constructed diagrams. 6. Systems Analyst may review and modify the diagrams.	
Preconditions	Candidate variables have been extracted and are ready.	
Postconditions	Causal loop diagrams are created and ready for analysis.	
Assumptions	The analyst has sufficient expertise to review and refine the generated diagrams.	

Figure 2. Use Case Scenario (UCS)

Based on these SPs, the resulting documents used as datasets are as follows:

(1) The researcher should access the data input form. IdVar4CL Application must display the page. The researcher should upload the document. IdVar4CL. Application must receive the document from the researcher.
(2) The researcher should access the preprocessing section. IdVar4CL Application must show the preprocessing interface. The researcher must select the uploaded documents. IdVar4CL Application should verify document availability. The researcher should start preprocessing. IdVar4CL Application must process the selected documents. IdVar4CL Application should apply case folding. IdVar4CL Application should tokenize the text. IdVar4CL Application should remove stopwords. IdVar4CL Application should apply stemming. IdVar4CL Application should save the processed text. IdVar4CL Application must notify preprocessing completion.
(3) The researcher should select document pairs for similarity calculation. IdVar4CL Application must verify selected documents are available. The researcher should initiate the similarity calculation. IdVar4CL Application must calculate similarity

using TF-IDF and semantic algorithms. The IdVar4CL Application should display similarity results to the researcher. A researcher may analyze the similarity scores.

(4) The researcher should select word pairs with similarity scores above 0.50. IdVar4CL Application must verify the selected word pairs. The researcher should initiate the extraction of candidate variables. IdVar4CL Application must extract candidate variables from the selected word pairs. The IdVar4CL Application should display the extracted candidate variables. The researcher may review and analyze the candidates.

(5) The Systems Analyst should select extracted variables. IdVar4CL Application must verify that the variables are available. The Systems Analyst should initiate diagram construction. IdVar4CL Application must generate causal loop diagrams from the variables. The IdVar4CL Application should display the constructed diagrams. Systems Analyst may review and modify the diagrams.

By applying a One-to-Many relationship across the 15 documents, a total of 75 variations of inter-document relationships are generated as datasets. The implementation of the One-to-Many relationship in this research is intended solely to maintain consistency in the linkage patterns and directional relationships among research artifacts.

It should be noted that the dataset used in this study is intentionally limited to the IdVar4CL case study and consists of 15 labeled documents, namely five Functional Requirements, five Use Cases, and five Steps Performed. These documents were expanded into 75 inter-document relationship pairs using a one-to-many relationship scheme. Therefore, the dataset represents a controlled case-study setting rather than a large-scale multi-project benchmark. This design allows the proposed pipeline to be evaluated in a focused, traceable manner, but it also limits the extent to which the model's performance can be generalized to other systems, domains, or documentation styles.

Based on the explanations of the three artifacts (FR, UCD, and UCS), the final documents used as datasets are defined as follows:

(1) D1 = Researcher must upload storytelling documents into the application interface.

(2) D2 = Researcher must initiate text preprocessing on uploaded documents, including case folding, tokenization, stopword removal, and stemming.

(3) D3 = Researcher must calculate similarity between document pairs using TF-IDF and semantic similarity algorithms.

(4) D4 = Researcher must extract candidate variables from word pairs with semantic similarity scores above 0.50.

(5) D5 = Systems Analyst must construct causal loop diagrams using the extracted variables.

(6) D6 = Upload Storytelling Document.

(7) D7 = Initiate Text Preprocessing.

(8) D8 = Calculate Similarity Documents.

(9) D9 = Extract Candidate Variables.

(10) D10 = Construct Causal Loop Diagram.

(11) D11 = The researcher should access the data input form. IdVar4CL Application must display the page. The researcher should upload the document. IdVar4CL Application must receive the document from the researcher.

(12) D12 = The researcher should access the preprocessing section. IdVar4CL Application must show the preprocessing interface. The researcher must select the uploaded documents. IdVar4CL Application should verify document availability. The researcher should start preprocessing. IdVar4CL Application must process the selected documents. IdVar4CL Application should apply case folding. IdVar4CL Application should tokenize the text. IdVar4CL Application should remove stopwords. IdVar4CL Application should apply stemming. IdVar4CL Application should save the processed text. IdVar4CL Application must notify preprocessing completion.

(13) D13 = The researcher should select document pairs for similarity calculation. IdVar4CL Application must verify selected documents are available. The researcher should initiate the similarity calculation. IdVar4CL Application must calculate similarity using TF-IDF and semantic algorithms. The IdVar4CL Application should display similarity results to the researcher. A researcher may analyze the similarity scores.

(14) D14 = Researcher should select word pairs with similarity scores above 0.50. IdVar4CL Application must verify the selected word pairs. The researcher should initiate the extraction of candidate variables. IdVar4CL Application must extract candidate variables from the selected word pairs. The IdVar4CL Application should display the extracted candidate variables. The researcher may review and analyze the candidates.

(15) D15 = Systems Analyst should select extracted variables. IdVar4CL Application must verify that the variables are available. The Systems Analyst should initiate diagram construction. IdVar4CL Application must generate causal loop diagrams from the variables. The IdVar4CL Application should display the constructed diagrams. Systems Analyst may review and modify the diagrams.

4. METHODOLOGY

This section presents an explanation of the pipeline executed across all stages of the research. Based on the data sources prepared in the dataset section, Figure 3 illustrates the stages and methods employed to achieve the research objectives and ensure successful outcomes. The pipeline is divided into four main activities:

(1) Input, consisting of SRS and SDD documents.

(2) Process, involving text processing applied to all SRS–SDD artifacts, which are collaboratively analyzed using BiLSTM and Machine Learning techniques.

(3) There is also a negative sample generation strategy, in which this experiment explores the formation of positive pairs and the criteria for determining negative pairs to identify the causes of false positives, which then form the basis for developing hard negatives in the next phase to improve precision.

(4) Output, consisting of documentation resulting from the Reverse Engineering activities, including the

USD, as well as recommendations for the further development of IdVar4CL.

Still referring to Figure 3, the following subsections describe each stage of the pipeline along with the corresponding target outcomes of this research:

(1) Input Stage: This stage involves collecting information for the USD of IdVar4CL for analysis. This activity is essential to ensure that the data used is complete, consistent, and relevant for the Reverse Engineering process.

(2) Process Stage: At this stage, the text preprocessing stage was configured to ensure that the transformation from raw SRS/SDD artifact text into model-ready input could be reproduced. Since the textual artifacts used in this experiment are written in English, the preprocessing pipeline applies English-oriented text normalization, stopword filtering, and stemming. The pipeline consists of five sequential steps: text normalization, case folding, tokenization, stopword removal, and stemming, followed by token reconstruction for downstream pair classification. During the Reverse Engineering process, a BiLSTM model is applied to analyze document relationships and directional links among the processed artifacts. The model construction consists of three phases: model building, training, and evaluation. The bidirectional capability of BiLSTM enables the model to capture contextual information from both directions within the text, thereby improving analytical accuracy. The final part of this stage is research validation using a Machine Learning workflow, which includes data splitting, model training, model testing, and performance evaluation. Recall indicates traceability completeness, precision indicates traceability correctness, F1-score balances completeness and correctness, accuracy indicates overall matrix consistency, and ROC-AUC supports threshold-based selection of traceability links. Before training the Siamese BiLSTM model, artifact pairs are constructed and labeled as linked or unlinked according to the defined negative-sample generation strategy.

(3) Negative Sample Generation Strategy: In this activity, there is a process of forming unconnected data samples (label 0) through a structural approach across artifact relationship groups in the initial stage, which is then developed using a hard negative sampling method based on high textual and semantic similarities outside the valid tracking path to minimize classification errors (false positives) and increase the accuracy (precision) of the model in recognizing actual traceability relationships. The labeling process follows the traceability assumption of the IdVar4CL artifacts. A pair is labeled as linked if it represents a valid semantic and directional relationship between artifact layers, such as FR–UC, UC–SP, or FR–SP. Conversely, a pair is labeled as unlinked if it does not support the same functional intention, scenario realization, or traceability path. The unlinked label is not assigned merely because two artifacts differ textually, but because the pair does not represent a valid requirement-to-design relationship within the USD structure.

(4) Output Stage: Several research targets are achieved in this stage. First, an updated USD incorporating the results of Reverse Engineering of UML artifacts analyzed using Text Processing and BiLSTM. This document provides a comprehensive representation of software requirements and design specifications derived from the Reverse Engineering process. Second, recommendations for improvement or further development based on the research findings are produced. These recommendations are intended to support stakeholders in making more informed decisions regarding future software development.

The output of the proposed pipeline is not limited to relationship prediction. Instead, the predicted links are transformed into a complete and traceable USD structure. In this context, completeness refers to the inclusion and preservation of all selected artifact layers—Functional Requirements, Use Cases, and Steps Performed—whereas traceability refers to the explicit representation of directional links among these layers. The generated USD therefore encodes FR→UC, UC→SP, and FR→SP mappings, enabling the construction of end-to-end FR→UC→SP paths. This design ensures that every reverse-engineered artifact can be traced back to its requirement origin and forward to its scenario-level realization, thereby strengthening the consistency between requirements specification and UML-based design documentation.

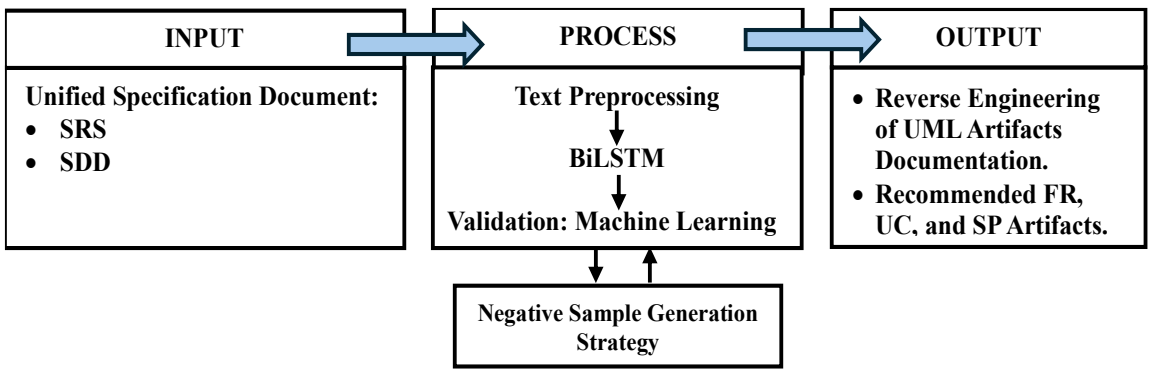

Figure 3. Methodology

4.1 Validation scope and generalization boundary

The validation conducted in this study is an internal case-study validation based on the IdVar4CL dataset. Stratified K-Fold validation was used to assess the stability of the Siamese BiLSTM model across the available labeled pairs while

maintaining balanced class proportions in each fold. However, because all documents originate from the same system context and follow relatively consistent writing patterns, the validation results should be interpreted as evidence of within-case performance rather than cross-project generalization. Cross-project or cross-system validation is required in future studies to evaluate whether the learned semantic linkage patterns remain robust when applied to different software domains, artifact structures, and requirement writing styles.

5. RESULT AND DISCUSSION

Referring to the previous chart in Figure 3, this section explains the implementation of all stages outlined in the chart. The explanation begins with input activities, which are USD exploration through Requirement Elicitation, followed by data preparation through Text Preprocessing. After Text Preprocessing, the Reverse Engineering activity is implemented using BiLSTM. The results of the Reverse Engineering are then validated using Machine Learning. The final activity is to present the output in the form of documentation and recommendations.

5.1 Input

The dataset used in this study is the IdVar4CL case study, in which SRS and SDD have been explored in previous studies [5-9]. In this input process, the USD format is prepared, with its artifacts derived from the SRS and SDD. As a limitation of the use of artifacts in this study, only three artifacts are used for Reverse Engineering, namely:

(1) Functional Requirements (FR) derived from Requirement Elicitation.
(2) Use Cases (UCs) derived from UCDs.
(3) Steps Performed (SP) derived from UCSs.

To capture the wide variety of relationships and directions among the three artifacts, we used a One-to-Many relationship. This relationship type is used solely to maintain consistency in the patterns of connections and directions between the research artifacts. The relationships used are as follows:

(1) Each FR is related to many Use Cases.
(2) Each Use Case is related to many Steps Performed (SP).
(3) Each FR is related to many SPs.

To illustrate the relationships among the three artifacts, Figure 4 shows how we obtained 75 interrelated documents. Furthermore, to perform Artifact Identification for this study, we labeled all documents from D1 to D15. Please see Table 3.

Table 3. Document labeling

No.	Artifact Identification	Labeling
1	FR-CL01	D1
2	FR-CL02	D2
3	FR-CL03	D3
4	FR-CL04	D4
5	FR-CL05	D5
6	UC-CL01	D6
7	UC-CL02	D7
8	UC-CL03	D8
9	UC-CL04	D9
10	UC-CL05	D10
11	SP-CL01	D11
12	SP-CL02	D12
13	SP-CL03	D13
14	SP-CL04	D14
15	SP-CL05	D15

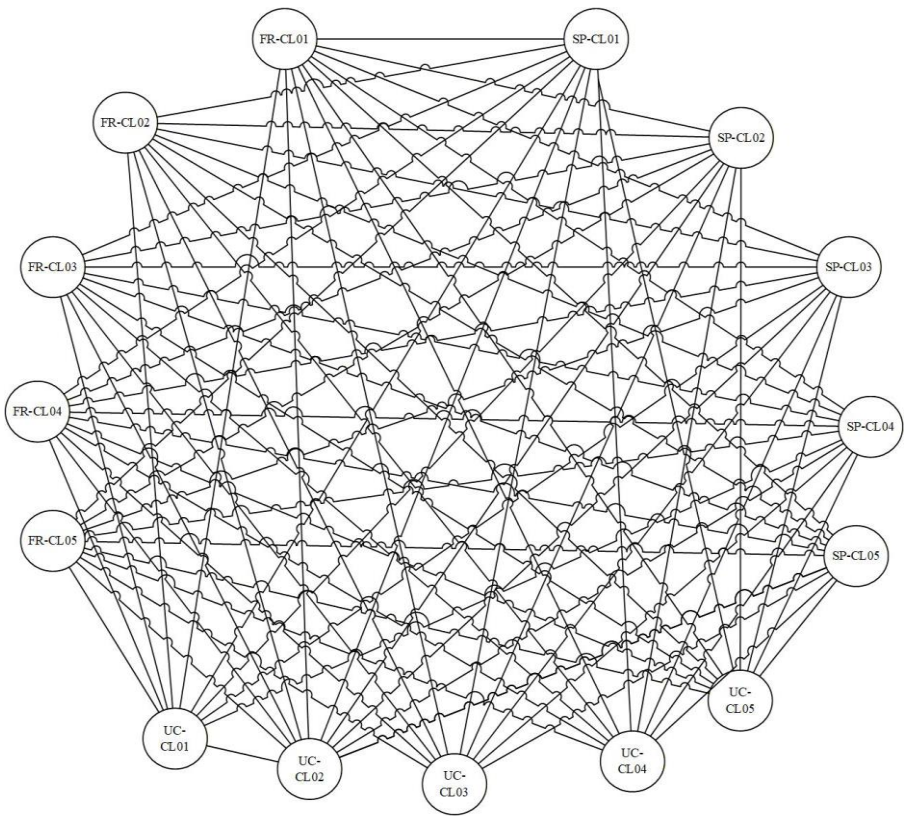

Figure 4. Illustration of one-to-many relationships and directional mapping between documents

5.2 Process: Text preprocessing results

In the Text Preprocessing stage, all documents D1–D15 are processed using a series of text-cleaning techniques, including case folding, tokenization, stopword removal, and stemming. This stage aims to reduce noise, simplify word variations, and improve the quality of text representation before entering the modeling stage.

The output generated at this stage is shown through before/after preprocessing examples for several documents (e.g., D2 and D12), as well as summary statistics such as average token length and vocabulary size after preprocessing. See Figure 5. In general, preprocessing helps reduce text complexity, especially in longer SP documents, allowing the model to focus more on relevant units of information.

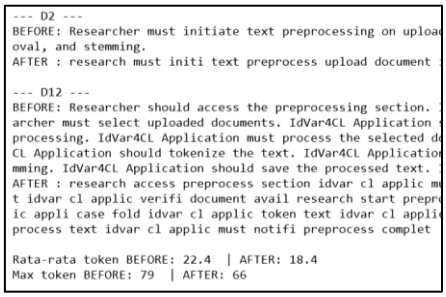

Figure 5. Example of text preprocessing results

5.3 Process: Bidirectional Long Short-Term Memory modeling results

The reverse-engineering of relationships between artifacts is implemented as a pair classification task with binary labels: 1 = linked and 0 = not linked. The model used is a Siamese BiLSTM, in which two input documents (A and B) are processed by a shared encoder (with shared weights) to produce a bidirectional vector representation. The pair representations are then combined via concatenation, absolute differences, and element-wise multiplication before being mapped to a connection probability using a sigmoid layer. We chose this approach because BiLSTM can capture bidirectional context in the requirement/scenario text, thereby supporting the learning of semantic relationships across artifacts more consistently than manual mapping. Please see Figure 6.

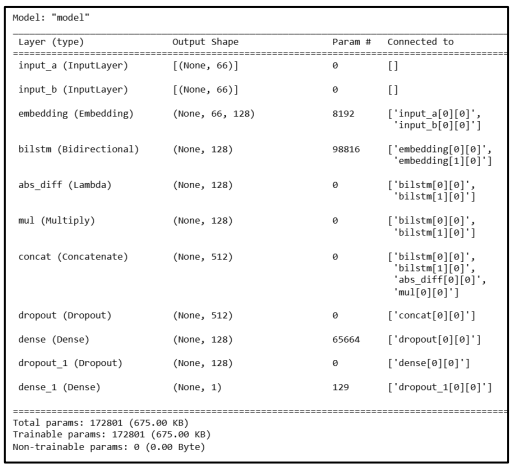

Figure 6. Siamese Bidirectional Long Short-Term Memory (BiLSTM) execution results

The model was trained using the hyperparameter configuration summarized in Table 4. Each fold was trained independently using the same architecture and training settings. The Siamese BiLSTM encoder used shared weights for both input documents, ensuring that FR, UC, and SP texts were encoded in the same semantic space. The output of the shared encoder was combined using concatenation, absolute difference, and element-wise multiplication before being passed to the dense classifier and sigmoid output layer. The sigmoid value was interpreted as the probability of a valid artifact link.

Table 4. Siamese Bidirectional Long Short-Term Memory (BiLSTM) hyperparameter configuration

Component	Parameter	Value
Input	Max sequence length	100
Input	Vocabulary size	5000
Embedding	Embedding dimension	100
BiLSTM	Number of layers	1
BiLSTM	Hidden units	64
Regularization	Dropout rate	0.3
Regularization	Recurrent dropout	0.2
Dense Layer	Units	64
Dense Layer	Activation	ReLU
Output Layer	Activation	Sigmoid
Loss Function	Loss	B.Cross-Entropy
Optimizer	Optimizer	Adam
Optimizer	Learning rate	0.001
Training	Batch size	16
Training	Epochs	50
Training	Early stopping	Patience = 5
Validation	Method	St.K-Fold (K = 5)
Threshold	Classification threshold	0.50 / 0.70

The training process minimized binary cross-entropy loss and evaluated model performance using accuracy, precision, recall, F1-score, and ROC-AUC in a stratified K-fold validation. This configuration ensures that the reported results reflect stable performance across different data splits and that the model's learning process can be consistently reproduced.

To improve reproducibility, the Siamese BiLSTM architecture, hyperparameters, training configuration, validation procedure, and experimental environment are explicitly reported. Each fold was trained independently using the same configuration, and the final results were reported as mean ± standard deviation across Stratified K-Fold validation.

5.4 Process: Validation results (Stratified K-Fold, K = 5)

Validation was performed using Stratified K-Fold (K = 5) to ensure balanced class proportions in each fold. Evaluation used Accuracy, Precision, Recall, F1-score, and ROC-AUC metrics. The evaluation results per fold are presented in Table 5 and summarized as mean ± standard deviation.

Table 5. Stratified K-Fold evaluation results (K = 5)

Fold	n-test	Accuracy	Precision	Recall	F1	ROC-AUC
1	30	1.0000	1.0000	1.000	1.0000	1.000
2	30	1.0000	1.0000	1.000	1.0000	1.000
3	30	0.9667	0.9375	1.000	0.9677	1.000
4	30	1.0000	1.0000	1.000	1.0000	1.000
5	30	0.9000	0.8333	1.000	0.9091	1.000

The summary of K-Fold results shows the following

average performance: Accuracy = 0.9733 ± 0.0435; Precision = 0.9542 ± 0.0728; Recall = 1.0000 ± 0.0000; F1 = 0.9754 ± 0.0396; and ROC-AUC = 1.0000 ± 0.0000. A recall value of 1.0 across all folds indicates that the model did not miss any positive (linked) pairs, thereby maintaining the required relationship coverage for traceability.

However, this result should be interpreted in the context of the relatively small and controlled IdVar4CL dataset. Since all document pairs are derived from the same system and share consistent terminology, the reported performance may be higher than expected in cross-project or cross-domain settings.

Based on the confusion matrix, Folds 1, 2, and 4 showed no misclassifications (15 TN and 15 TP). Fold 3 had 1 false positive (14 TN, 1 FP, 15 TP), while Fold 5 had 3 false positives (12 TN, 3 FP, 15 TP). No false negatives were found across all folds, which is consistent with Recall=1.0. See Figure 7.

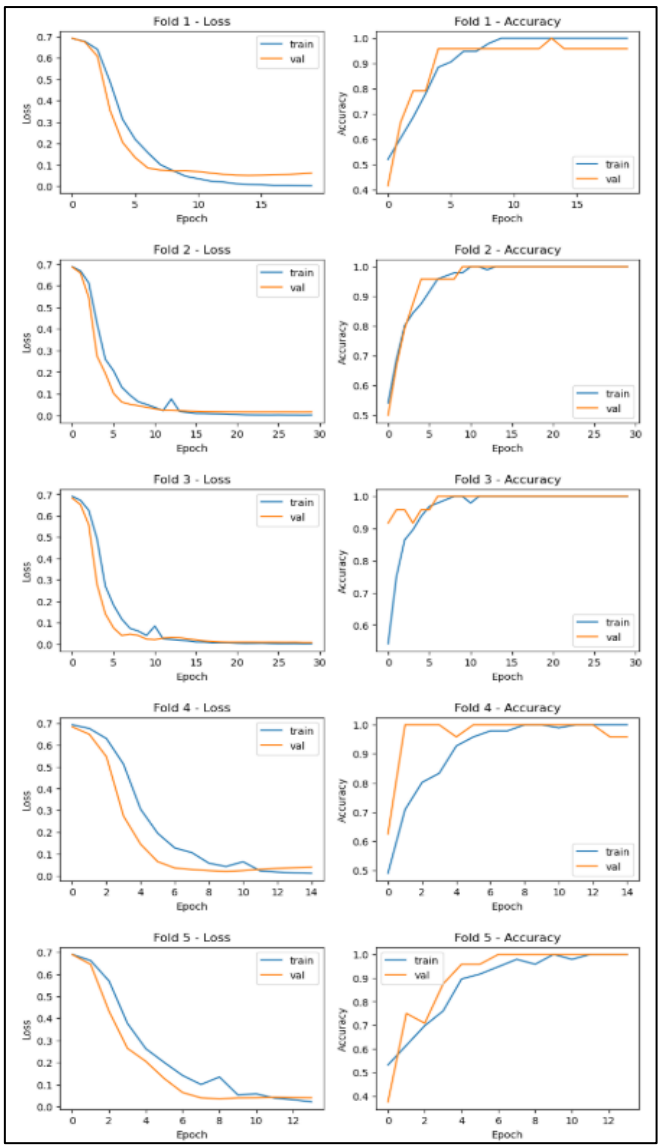

Figure 7. Learning curve (loss and accuracy) per fold

The evaluation metrics directly affect the quality of the generated USD traceability matrix. Accuracy reflects the overall consistency of linked and unlinked pair classification, indicating whether the matrix generally represents the expected artifact relationships. Precision reflects the correctness of predicted links; therefore, lower precision may introduce false traceability links and cause over-linking among FR, UC, and SP artifacts. Recall reflects traceability completeness, because false negatives may remove valid FR→UC or UC→SP links and break the continuity of FR→UC→SP paths. F1-score balances these two concerns by indicating whether the model can maintain both link correctness and link coverage. ROC-AUC reflects the model's ability to distinguish linked and unlinked pairs across thresholds, which is useful for future probability-based construction of traceability matrices.

In this study, the perfect recall across all folds indicates that no valid linked pairs were missed. From a traceability perspective, this is important because it indicates that the generated USD maintains complete coverage of the expected artifact relationships in the IdVar4CL case study. However, the presence of false positives in Fold 3 and Fold 5 indicates that some unlinked pairs were incorrectly classified as linked. These errors may not break traceability coverage, but they may reduce traceability precision by adding unnecessary or incorrect links to the USD matrix. Therefore, the main risk identified in this study is not missing traceability paths, but over-linking caused by false positives.

With high, stable validation performance, predicted interdocument relationships can serve as a measurable basis for constructing cross-document artifact integration. In the following subsections, these findings are translated into two main research outputs:

(1) Updated USD representing FR–UC–SP traceability.
(2) Recommendations for improvement/development derived from error patterns (false positives) and areas of ambiguity in document pairs.

5.5 Negative sample generation strategy

This study formulates relationship inference as a binary pair classification task, where artifact pairs are labeled 1 for linked and 0 for unlinked. Positive samples represent valid directional relationships among FR, UC, and SP, namely FR–UC, UC–SP, and FR–SP, while negative samples represent pairs without valid semantic or directional links in the IdVar4CL traceability structure. Negative samples are created to help the model distinguish truly linked pairs from pairs with similar terminology but different requirement-to-design meanings. They are generated by pairing artifacts across different relationship groups while preserving the same artifact-type combinations, such as mismatched FR–UC, FR–SP, or UC–SP pairs. This strategy prevents the model from relying solely on artifact-level differences and encourages semantic-compatibility evaluation between paired documents.

However, random or structurally generated negative samples may still be insufficient when textual artifacts share overlapping domain terms, actors, or action verbs. In the IdVar4CL dataset, many documents contain repeated terms such as "Researcher," "IdVar4CL Application," "document," "preprocessing," "similarity," "variables," and "diagram." As a result, some unlinked pairs may appear lexically similar to linked pairs, causing the model to classify them as positive. This phenomenon is reflected in the validation results, where false positives appear in Fold 3 and Fold 5. Therefore, negative sample construction is treated as an important factor affecting precision and the reliability of traceability generation.

To address this issue, a hard negative sampling strategy is proposed as an extension of the current negative sample generation process. Hard negatives refer to unlinked artifact

pairs that are lexically or semantically similar to positive pairs but differ in their actual traceability meaning. These pairs are more challenging than random negatives because they require the model to learn fine-grained semantic distinctions rather than relying on surface-level word overlap. In future experiments, hard negatives will be generated by selecting unlinked pairs with high textual similarity scores, shared actors, shared domain terms, or similar verb-object structures, while ensuring that they do not belong to the same FR→UC→SP traceability path.

Operationally, hard negative samples can be constructed using the following criteria:

(1) Lexical-overlap negatives: unlinked pairs that share important domain terms but belong to different artifact paths.
(2) Actor-similarity negatives: unlinked pairs that involve the same actor, such as "Researcher" or "Systems Analyst," but describe different functional intentions.
(3) Verb-object similarity negatives: unlinked pairs that contain similar action structures, such as "select document," "process document," or "display result," but refer to different use case contexts.
(4) High-similarity score negatives: unlinked pairs whose TF-IDF or semantic similarity score is above a predefined threshold but are not part of the ground-truth traceability path.
(5) Cross-scenario negatives: UC–SP pairs in which the scenario steps belong to a different use case, even though they share similar terminology.

By incorporating these hard negatives, the model is expected to reduce false positives and improve precision, especially when unrelated artifacts exhibit high lexical or structural similarity. Refer to Table 6 for the implemented strategic plan.

Table 6. Negative sample generation strategy

Negative Sample	Construction Rule	Example Condition	Purpose
Random negative	Pair artifacts that do not belong to the same traceability path	FR-CL01 paired with an unrelated UC or SP	Provides basic unlinked examples
Same-type structure negative	Preserve FR–UC, UC–SP, or FR–SP format but assign unrelated pairs	UC from one scenario paired with SP from another scenario	Prevents the model from relying only on artifact type
Lexical-overlap hard negative	Select unlinked pairs sharing domain terms	Both pairs contain "document" or "preprocessing" but refer to different functions	Tests whether the model can distinguish meaning beyond word overlap
Actor-based hard negative	Select unlinked pairs with the same actor	"Researcher" appears in both artifacts but the actions differ	Reduces actor-name bias
Verb-object hard negative	Select unlinked pairs with similar action-object patterns	"select document" versus "upload document"	Forces finer semantic discrimination
High-similarity hard negative	Select unlinked pairs with high TF-IDF or semantic similarity	Similarity score above threshold but no valid traceability relation	Reduces false positives caused by surface similarity
Cross-scenario hard negative	Pair UC with SP from another Use Case Scenario	UC-CL02 paired with SP-CL03	Improves UC→SP path discrimination

5.6 False positive analysis and negative sample quality

The validation results show that the model achieved perfect recall across all folds, meaning that no positive linked pairs were missed. However, false positives occurred in Fold 3 and Fold 5. This indicates that the model occasionally classified unlinked pairs as linked. Such errors are important in the context of traceability generation because false positives may introduce incorrect FR→UC, UC→SP, or FR→SP links into the updated USD. If not controlled, these incorrect links may cause over-linking and reduce the precision of the generated FR→UC→SP paths.

The presence of false positives suggests that the current negative sample construction may not be sufficiently challenging. Several IdVar4CL artifacts share similar actors, domain terms, and sentence structures. For example, multiple documents refer to the "Researcher," "IdVar4CL Application," "documents," "similarity," "variables," and "diagrams." Such lexical overlap may cause the Siamese BiLSTM model to assign high similarity to unlinked pairs, especially when the negative examples do not include enough semantically close but incorrect pairs. Therefore, the false positives are likely related not only to model behavior but also to the difficulty level and representativeness of the negative samples.

To improve the model's robustness, future training should incorporate hard negative sampling. In this strategy, negative samples are intentionally selected from unlinked pairs that are lexically or structurally similar to positive pairs. This can include pairs with shared actors, similar verbs, common domain terms, or high textual similarity scores that do not belong to the same traceability path. By exposing the model to more difficult unlinked examples, hard negative sampling is expected to improve precision and reduce the risk of over-linking in the generated USD.

5.7 Process: Impact of evaluation metrics on FR→UC→SP traceability

The main objective of the evaluation is not only to assess whether the Siamese BiLSTM model can correctly classify artifact pairs, but also to determine whether the predicted links can support the construction of a reliable USD traceability matrix. In this study, the traceability matrix is constructed from predicted relationships among FR–UC, UC–SP, and FR–SP pairs. These relationships are then interpreted as directional traceability paths connecting Functional Requirements to Use Cases and Steps Performed. From this perspective, recall is the most critical metric for traceability completeness. If the model produces false negatives, valid links may be omitted from the USD. For example, a missed FR→UC link would prevent a functional requirement from being connected to its corresponding use case, while a missed UC→SP link would prevent the use case from being elaborated into scenario-level steps. Such missing links could break the end-to-end FR→UC→SP path. Therefore, the recall value of 1.0000

across all folds indicates that the model preserved complete traceability coverage for the linked pairs in the IdVar4CL dataset.

Precision is equally important for traceability correctness. Although high recall ensures that valid links are not missed, false positives may introduce incorrect links into the traceability matrix. In the context of USD construction, false positives may result in over-linking, where an FR is connected to an unrelated UC or SP, or a UC is connected to scenario steps from another context. This may reduce the interpretability and reliability of the generated USD. Therefore, the observed false positives in Fold 3 and Fold 5 indicate that negative sample construction and semantic validation should be strengthened to improve precision.

F1-score provides a balanced view of traceability quality by combining precision and recall. A high F1-score indicates that the model can maintain both completeness and correctness of trace links. In this study, the average F1-score of 0.9754 ± 0.0396 suggests that the generated USD traceability matrix achieves a strong balance between preserving valid links and avoiding incorrect links, although remaining false positives still require further improvement.Accuracy provides a general indication of matrix consistency, but it is not sufficient as the sole metric for traceability construction. In traceability tasks, the costs of false negatives and false positives are different.

False negatives may break traceability paths, while false positives may create misleading paths. Therefore, accuracy must be interpreted together with precision, recall, and F1-score. ROC-AUC indicates the model's ability to distinguish linked and unlinked pairs across different thresholds. This is relevant for future USD construction because traceability matrices can be generated not only as binary matrices but also as probability-based matrices. In such a setting, ROC-AUC can guide threshold calibration to determine which predicted links should be included in the USD. This would allow stakeholders to balance traceability completeness and correctness according to project needs.

Table 7. Impact of classification errors on Unified Specification Document (USD) traceability

Error Type	Classification Meaning	Effect on Traceability	Effect on FR→UC→SP Path	Risk
True Positive	Valid linked pair correctly predicted as linked	Correct link is included	Valid path is preserved	Desired outcome
True Negative	Invalid pair correctly predicted as unlinked	Incorrect link is excluded	Invalid path is avoided	Desired outcome
False Positive	Invalid pair predicted as linked	Extra incorrect link is added	May create misleading FR →UC→SP path	Over-linking
False Negative	Valid pair predicted as unlinked	Required link is omitted	May break a valid FR→UC →SP path	Broken traceability

In practical terms, the two main error types have different effects on the USD. A false negative removes a valid traceability link. For example, if FR-CL02 is not linked to its corresponding preprocessing use case, the USD would fail to show how the requirement is realized in the design artifact. This reduces traceability completeness. Conversely, a false positive adds an invalid traceability link. For example, if a preprocessing-related FR is incorrectly linked to an unrelated diagram construction scenario, the USD would contain an incorrect FR→UC→SP path. This reduces traceability correctness and may mislead stakeholders during requirements validation or design review. Please see Table 7.

5.8 Output 1: Updated Unified Specification Document

The updated USD is constructed based on the artifact-pair classifier's evaluation results by transforming predicted semantic links into explicit traceability relations among FR, UC, and SP. True positive links are included as valid traceability mappings, while true negative pairs are excluded. False positives may reduce precision by adding incorrect links, whereas false negatives may reduce traceability completeness by omitting required links. Therefore, recall represents traceability coverage, precision reflects traceability correctness, and the F1-score serves as a combined indicator of overall USD traceability quality. As the final artifact of the proposed reverse-engineering pipeline, the updated USD provides more than a prediction report; it serves as an integrative documentation artifact with complete, directional traceability. It explicitly represents FR→UC, UC→SP, and FR→SP mappings derived from document pairs labeled linked (1), enabling end-to-end FR→UC→SP paths. These paths show how functional requirements are realized through use cases and operationalized into scenario-level steps, supporting both forward and backward traceability, minimizing semantic drift, and strengthening specification-design consistency.

Example traceability path (illustrative): D1 (FR-CL01) → {D6..D10} (UC) → {D11..D15} (SP).

Table 8. Traceability matrix FR × UC

FR\ UC	UC-CL01	UC-CL02	UC-CL03	UC-CL04	UC-CL05
FR-CL01	0.95 ✓	0.42 ×	0.38 ×	0.35 ×	0.30 ×
FR-CL02	0.44 ×	0.96 ✓	0.41 ×	0.37 ×	0.33 ×
FR-CL03	0.36 ×	0.45 ×	0.94 ✓	0.58 ?	0.32 ×
FR-CL04	0.34 ×	0.40 ×	0.57 ?	0.93 ✓	0.39 ×
FR-CL05	0.29 ×	0.36 ×	0.42 ×	0.47 ×	0.97 ✓

Table 9. Traceability matrix UC × SP

UC\ SP	SP-CL01	SP-CL02	SP-CL03	SP-CL04	SP-CL05
UC-CL01	0.94 ✓	0.40 ×	0.36 ×	0.34 ×	0.31 ×
UC-CL02	0.41 ×	0.96 ✓	0.45 ×	0.39 ×	0.35 ×
UC-CL03	0.38 ×	0.44 ×	0.95 ✓	0.59 ?	0.33 ×
UC-CL04	0.35 ×	0.39 ×	0.56 ?	0.93 ✓	0.41 ×
UC-CL05	0.32 ×	0.37 ×	0.43 ×	0.46 ×	0.97 ✓

Table 10. Traceability matrix FR × SP

FR\ SP	SP-CL01	SP-CL02	SP-CL03	SP-CL04	SP-CL05
FR-CL01	0.92 ✓	0.41 ×	0.37 ×	0.34 ×	0.30 ×
FR-CL02	0.43 ×	0.95 ✓	0.44 ×	0.39 ×	0.35 ×
FR-CL03	0.39 ×	0.45 ×	0.93 ✓	0.60 ?	0.33 ×
FR-CL04	0.36 ×	0.40 ×	0.58 ?	0.91 ✓	0.42 ×
FR-CL05	0.31 ×	0.37 ×	0.46 ×	0.48 ×	0.96 ✓

Note: ✓ indicates accepted links ($p \geq 0.70$), ? indicates borderline links ($0.50 \leq p < 0.70$), and × indicates rejected links ($p < 0.50$). These matrices are based on probability predictions from the Siamese BiLSTM model and reflect its confidence in identifying valid and invalid artifact relationships, rather than assuming full connectivity.

To formally present the mapping, Tables 8–10 show probability-based traceability matrices between artifacts. Each cell represents the predicted probability of a semantic link generated by the Siamese BiLSTM model and is interpreted using thresholds: ✓ ($p \geq 0.70$) for accepted links, ? ($0.50 \leq p < 0.70$) for borderline links, and × ($p < 0.50$) for rejected links. This approach reflects the model's ability to distinguish valid and invalid relationships rather than assuming full connectivity among artifacts.

The evaluation results show that the generated traceability matrices achieve strong within-case coverage, with no false negatives across all folds, indicating that all expected linked relationships are preserved in the IdVar4CL setting. However, false positives suggest that some predicted links still require semantic validation before inclusion in the USD. Thus, the proposed pipeline should be viewed as a semi-automated mechanism for constructing traceability, where model predictions generate candidate links and human or rule-based validation confirms borderline cases. Collectively, these matrices support end-to-end FR→UC→SP path construction, enabling stakeholders to trace how requirements relate to use case behavior and scenario steps while aligning SRS/SDD artifacts with UML-based documentation.

5.9 Output 2: Recommendations for improvement/ development

Recommendations are based on validation results and error analysis. Despite excellent recall, several false positives occurred in Fold 3 and Fold 5, reducing precision. This finding indicates that similarities in sentence structure or terminology between documents can lead the model to treat unrelated pairs as similar. Based on these findings, recommendations include:

(1) Standardizing domain terminology and requirement/ scenario writing patterns to reduce ambiguity.

(2) Applying hard negative sampling in follow-up experiments by constructing unlinked artifact pairs that are lexically or structurally similar to linked pairs. These hard negatives may include pairs with shared actors, similar domain terms, similar verb-object patterns, or high TF-IDF/semantic similarity scores, yet remain outside the valid FR→UC→SP traceability path. This strategy is expected to reduce false positives and improve precision by forcing the model to distinguish true semantic traceability from superficial textual similarity.

(3) Implementing a human-in-the-loop to review pairs with borderline probability as part of an audit trail and trustworthiness assessment.

(4) Metric-based USD validation interprets recall as traceability coverage, precision as traceability correctness, F1-score as their balance, and ROC-AUC as the basis for threshold calibration to decide whether FR→UC→SP paths should be accepted, reviewed, or rejected.

5.10 Dataset size, generalization, and threats to validity

Although the proposed Siamese BiLSTM pipeline achieved strong validation results, the dataset size should be considered when interpreting the model performance. The current experiment uses 15 labeled documents from the IdVar4CL case study, consisting of five Functional Requirements, five Use Cases, and five Steps Performed. These documents were transformed into 75 inter-document relationship pairs through a one-to-many relationship scheme. While this setting is sufficient to demonstrate the feasibility of the proposed reverse-engineering pipeline, it remains relatively small for drawing broad generalization claims.

The small dataset size may affect the reported model performance in several ways. First, the high accuracy, recall, F1-score, and ROC-AUC values may partly reflect the controlled structure of the IdVar4CL artifacts, where the documents follow consistent terminology, writing patterns, and artifact relationships. Second, because all data originate from the same system context, the model may learn case-specific semantic patterns rather than generalizable cross-domain relationships. Third, the one-to-many relationship construction increases the number of training pairs, but it does not fully replace the diversity that would be obtained from additional independent projects or systems. Therefore, the current validation results should be interpreted as within-case evidence rather than proof of general applicability.

Another potential impact is the risk of overestimating model robustness. Stratified K-Fold validation helps evaluate performance stability across the available pairs, but the folds are still derived from the same underlying document set. As a result, lexical overlap and repeated domain-specific terms across FR, UC, and SP artifacts may make the classification task easier than in a real cross-project scenario. This may explain why the model achieved perfect recall across all folds and ROC-AUC values of 1.0000. Although this result is beneficial for maintaining traceability coverage, it should be interpreted carefully because external validation has not yet been conducted.

To mitigate this limitation, future work should expand the dataset to include multiple software systems, diverse application domains, and more varied SRS/SDD writing styles. Cross-project validation can be performed by training the model on one or more systems and testing it on another system. Cross-system validation can also be used to evaluate whether the model can infer FR→UC→SP paths in unseen documentation contexts. In addition, future experiments should include more rigorous negative pair construction, hard negative sampling, and probability threshold calibration to reduce false positives and improve generalization. These extensions are necessary before the proposed pipeline can be claimed as broadly applicable beyond the IdVar4CL case study.

6. CONCLUSION AND FUTURE WORK

This study proposes a reproducible text-driven reverse-engineering pipeline for UML-related artifacts by combining standard text preprocessing and a Siamese BiLSTM architecture to infer semantic linkages among FR, UC, and SP. Using the IdVar4CL USD as a case study, the task is formulated as binary-pair classification across 15 labeled documents (D1–D15), yielding 75 inter-document relationships. Under Stratified K-Fold validation ($K = 5$), the model achieved strong performance, with accuracy of 0.9733 ± 0.0435, precision of 0.9542 ± 0.0728, recall of 1.0000 ± 0.0000, F1-score of 0.9754 ± 0.0396, and ROC-AUC of 1.0000 ± 0.0000. The perfect recall confirms that no positive linkages were missed, supporting complete end-to-end traceability in documentation-to-design alignment..

Beyond prediction, this study contributes by converting

semantic linkage results into a complete and traceable USD. The updated USD represents FR→UC, UC→SP, and FR→SP mappings to generate end-to-end FR→UC→SP paths linking requirements, use cases, and scenario-level steps. Thus, the approach serves as a reverse-engineering mechanism to produce traceable specification-design documentation, reduce semantic drift, and improve consistency between requirements and UML-related artifacts..

This study has limitations in dataset size, generalization, and negative sample construction. The evaluation used 15 labeled IdVar4CL documents expanded into 75 inter-document pairs, providing feasibility evidence within the IdVar4CL case study but not broad generalizability across diverse SRS/SDD documents, projects, and domains. False positives across multiple folds suggest that the current negative sampling strategy may not capture difficult, unlinked pairs with high lexical or structural similarity. Future work should use larger cross-project and cross-system datasets, hard negative sampling, threshold calibration, multi-relation classification for unrelated, FR→UC, UC→SP, and FR→SP relations, and ablation studies on preprocessing, stopword removal, stemming, modal-verb preservation, Siamese architecture, BiLSTM bidirectionality, hard negative sampling, and threshold calibration to improve generalization, reduce false positives, and strengthen FR→UC→SP path reliability in the USD.

ACKNOWLEDGMENT

This work was supported by Directorate of Research and Community Service (PPM Tel-U) at Telkom University Bandung 40257. Research Implementation Agreement Letter No. 413/LIT06/PPM-LIT/2025.

REFERENCES

[1] Alenezi, M., Akour, M. (2025). AI-driven innovations in software engineering: A review of current practices and future directions. Applied Sciences, 15(3): 1344. https://doi.org/10.3390/app15031344

[2] Martins, H.F., Junior, A.C.d.O., Canedo, E.D., Kosloski, R.A.D., Paldês, R.Á., Oliveira, E.C. (2019). Design thinking: Challenges for software requirements elicitation. Information, 10(12): 371. https://doi.org/10.3390/info10120371

[3] Priyadi, Y. (2025). Software requirement specification compatibility design method through use case diagram artifacts using text mining on akuonline learning application. Ingénierie des Systèmes d'information, 30(5): 1123-1134. https://doi.org/10.18280/isi.300502

[4] Koç, H., Erdoğan, A.M., Barjakly, Y., Peker, S. (2021). UML diagrams in software engineering research: A systematic literature review. Proceedings, 74(1): 13. https://doi.org/10.3390/proceedings2021074013

[5] Seba, M.D., Priyadi, Y., Darwiyanto, E. (2024). Software development for text processing in mapping architecturally significant requirements towards quality attributes. In 2024 10th International Conference on Smart Computing and Communication (ICSCC), Bali, Indonesia, pp. 394-400. https://doi.org/10.1109/ICSCC62041.2024.10690818

[6] Laksana, M.A.I., Priyadi, Y., Wibowo, Y.F.A. (2025). Formation of use case scenario based on use case diagram using text semantics for IdVar4CL application development. In 2025 10th International Conference on Signal Processing and Communication (ICSC), Noida, India, pp. 415-420. https://doi.org/10.1109/ICSC64553.2025.10968713

[7] Fatimatuzzahra, N., Priyadi, Y., Nurtantyana, R. (2025). Functional and non-functional requirement (FR and NFR) formation based on requirement elicitation using text semantics in IdVar4CL artifact. In 2025 10th International Conference on Signal Processing and Communication (ICSC), Noida, India, pp. 421-426. https://doi.org/10.1109/ICSC64553.2025.10968323

[8] Priyadi, Y., Kusumahadi, K., Lyanda, P.S. (2022). IdVar4CL: Causal loop variable identification method for systems thinking based on text mining approach. International Journal of Fuzzy Logic and Intelligent Systems, 22(4): 373-381. https://doi.org/10.5391/ijfis.2022.22.4.373

[9] Rachmanda, M.A., Priyadi, Y., Fathoni, M.F. (2025). Object identification for sequence diagram based on use case scenario using text semantic: Case study of IdVar4CL artefact. In 2025 5th International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS), Yogyakarya, Indonesia, pp. 125-130. https://doi.org/10.1109/ICE3IS66769.2025.11280707

[10] Xu, K., Feng, Y., Li, Q., Dong, Z., Wei, J. (2025). Survey on terminology extraction from texts. Journal of Big Data, 12(1): 1-40. https://doi.org/10.1186/s40537-025-01077-x

[11] Merrouni, Z.A., Frikh, B., Ouhbi, B. (2023). EXABSUM: A new text summarization approach for generating extractive and abstractive summaries. Journal of Big Data, 10(1): 1-34. https://doi.org/10.1186/s40537-023-00836-y

[12] Pais, S., Cordeiro, J., Jamil, M.L. (2022). NLP-based platform as a service: A brief review. Journal of Big Data, 9(1): 1-26. https://doi.org/10.1186/s40537-022-00603-5

[13] Shorten, C., Khoshgoftaar, T.M., Furht, B. (2021). Text data augmentation for deep learning. Journal of Big Data, 8(1): 1-34. https://doi.org/10.1186/s40537-021-00492-0

[14] Chen, J., Zhang, T., Yan, Z., Zheng, Z., Zhang, W., Zhang, J. (2025). Attention-based BiLSTM with positional embeddings for fake review detection. Journal of Big Data, 12(1): 1-23. https://doi.org/10.1186/s40537-025-01130-9

[15] Zhang, H., Chen, C., Ran, P., Yang, K., Liu, Q., Sun, Z., Chen, J., Chen, J. (2024). A multi-dimensional hierarchical evaluation system for data quality in trustworthy AI. Journal of Big Data, 11(1): 1-26. https://doi.org/10.1186/s40537-024-00999-2

[16] Shamsuddin, R., Tabrizi, H.B., Gottimukkula, P.R. (2025). Towards responsible AI: An implementable blueprint for integrating explainability and social-cognitive frameworks in AI systems. AI Perspectives & Advances, 7(1): 1-23. https://doi.org/10.1186/s42467-024-00016-5

[17] Adadi, A., Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). IEEE Access, 6: 52138-52160. https://doi.org/10.1109/access.2018.2870052

[18] Yahya, M., Breslin, J.G., Ali, M.I. (2021). Semantic web and knowledge graphs for industry 4.0. Applied

Sciences, 11(11): 5110. https://doi.org/10.3390/app11115110
[19] Li, Y., Wang, T., Yu, L., Pan, Z. (2025). Fus: Combining semantic and structural graph information for binary code similarity detection. Electronics, 14(19): 3781. https://doi.org/10.3390/electronics14193781
[20] Cohen, R., David, R., Yger, F., Rossi, F. (2025). Identifying obfuscated code through graph-based semantic analysis of binary code. Applied Network Science, 10(1): 1-24. https://doi.org/10.1007/s41109-025-00733-8
[21] Rahman, M., Watanobe, Y., Nakamura, K. (2021). A bidirectional LSTM language model for code evaluation and repair. Symmetry, 13(2): 247. https://doi.org/10.3390/sym13020247
[22] Bai, J., Zhu, W., Liu, S., Ye, C., Zheng, P., Wang, X. (2025). A Temporal Convolutional Network–Bidirectional Long Short-Term Memory (TCN-BiLSTM) prediction model for temporal faults in industrial equipment. Applied Sciences, 15(4): 1702. https://doi.org/10.3390/app15041702
[23] Du, S., Wang, W., Fu, H., Wan, X. (2023). Fault detection and state estimation in automatic control. Applied Sciences, 13(23): 12936. https://doi.org/10.3390/app132312936
[24] Kutay, E., Yener, A. (2025). Harnessing the power of pre-trained models for efficient semantic communication of text and images. Entropy, 27(8): 813. https://doi.org/10.3390/e27080813
[25] Patil, R., Gudivada, V. (2024). A review of current trends, techniques, and challenges in Large Language Models (LLMs). Applied Sciences, 14(5): 2074. https://doi.org/10.3390/app14052074
[26] Nagwani, N.K. (2015). Summarizing large text collection using topic modeling and clustering based on MapReduce framework. Journal of Big Data, 2(1): 6. https://doi.org/10.1186/s40537-015-0020-5
[27] Kowald, D., Scher, S., Pammer-Schindler, V., Müllner, P., et al. (2024). Establishing and evaluating trustworthy AI: Overview and research challenges. Frontiers in Big Data, 7: 1467222. https://doi.org/10.3389/fdata.2024.1467222