

RESEARCH ARTICLE



NS-RAD: A Neuro-Symbolic Framework for Conflict and Ambiguity Detection in Natural-Language Requirements

Anis Zarrad¹, Jaber Jemai^{2,*} , Yassine Daadaa³ and Danyal Abbasi¹

¹*School of Computer Science, University of Birmingham, UAE*

²*Faculty of Data Science, Higher Colleges of Technology, UAE*

³*College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Saudi Arabia*

Abstract: High-quality software requirements are critical for delivering dependable systems on time and within budget, yet natural-language Software Requirements Specifications (SRS) frequently introduce defects, rework, and project delays. We introduce NS-RAD, a multi-layer neuro-symbolic feedback-driven framework for automated detection of two common requirement anomalies: (i) conflicts, where statements cannot be simultaneously satisfied, and (ii) ambiguities, where vague or underspecified language allows for divergent interpretation. It consists of four parts: requirement ingestion and normalization, multi-view semantic and linguistic encoding, dual analysis for conflict and ambiguity detection, and a feedback-driven adaptation layer. We show that NS-RAD achieves high accuracy, cross-dataset generalization, and practical utility on several public SRS datasets. The framework provides detection performance on par with representative recent approaches, calibrated risk assessment, explicit evidence traces, and actionable guidance for analysts. The results show that combining transformer-based learning with linguistically informed rules in a feedback-driven architecture is a powerful and scalable approach to enhance clarity, consistency, and reliability of industrial requirements engineering.

Keywords: software requirements quality assurance, natural language processing, neuro-symbolic AI

1. Introduction

Requirements Engineering (RE) is a major driver of project success, as defects introduced during requirements elicitation propagate downstream and incur significant cost and schedule overheads. Empirical studies [1, 2] show that a large percentage of software defects are introduced during the requirements phase, and their late discovery directly contributes to rework and project delays. For many decades, formal modeling and specification templates have been studied, but industrial practice still heavily relies on natural-language Software Requirements Specifications (SRS) documents, which are prone to anomalies.

Two types of anomalies are especially harmful in natural-language requirements: conflicts and ambiguities. Ambiguous requirements are expressed in vague, subjective, or context-dependent language that may lead to divergent stakeholder interpretations, and conflicting requirements cannot be satisfied at the same time. Doing this manually for large repositories is time-consuming, mentally exhausting, and heavily reliant on the expertise of the analyst.

Previous approaches include rule-based and formal modeling approaches, similarity-based machine learning (ML), and transformer-based semantic analysis. Rule-based techniques [3] provide accurate analyses for well-structured domains but do not scale well for heterogeneous SRS documents. Similarity-based methods such as the Supervised Semantic Similarity-based Conflict Detection Algorithm (S3CDA) [4] detect conflicting pairs with sentence embedding and entity overlap but depend on hand-tuned thresholds and do not model interactions between semantic and linguistic cues. Existing ambiguity detectors [5] rely on static keyword lists or shallow syntactic patterns, which either overproduce false positives or miss subtle context-dependent ambiguities.

To address these limitations, this paper presents the Neuro-Symbolic Requirements Anomaly Detector (NS-RAD), a multi-layer, feedback-driven system that automatically detects conflicts and ambiguities in English-language SRS documents. NS-RAD handles functional and nonfunctional requirements at the sentence level. It detects conflicts between pairs of sentences and intra-sentence ambiguity. Conflicts are identified using transformer embeddings, Natural Language Inference (NLI) signals, and engineered features like negation patterns, numeric-range checks, and role overlap. We employ a linguistically motivated rule engine and a lightweight classifier for ambiguity detection that

*Corresponding author: Jaber Jemai, Faculty of Data Science, Higher Colleges of Technology, UAE. Email: jjemai@hct.ac.ae

serves as a filter for spurious activations while preserving interpretability. Decision logic combines learned feature interactions with explicit rule triggers. An automatic label-driven calibration procedure is used to derive and update the operating thresholds for anomaly, requires-check, and neutral decisions.

The main research problem addressed in this paper is how to automatically detect conflicting and ambiguous requirements with sufficient accuracy, scalability, and practical utility to reduce manual review effort in industrial RE practice. The ability of NS-RAD in correctly differentiating conflicting and ambiguous requirements in heterogeneous datasets is assessed in this study. It also shows the performance of NS-RAD by comparing it with the state-of-the-art methods in terms of accuracy, robustness, interpretability, and scalability.

The proposed framework (NS-RAD) makes four key contributions over the previous work. First, it combines transformer embeddings with engineered linguistic and semantic features to capture NLI signals, negation, lexical overlap, numeric-range cues, and role overlap, enabling learned interactions beyond similarity-threshold methods. Second, it extracts empirically known ambiguity patterns into lexical, syntactic, and contextual rules and a lightweight classifier for filtering spurious activations. Third, it integrates conflict and ambiguity detectors into a single reproducible workflow with a fully specified automatic calibration procedure. Fourth, it is evaluated on public benchmarks of SRS and compared to representative recent methods, in terms of robustness, interpretability, and scalability.

The rest of this paper is organized as follows. Section 2 discusses related work in the area of requirements anomaly detection. Section 3 describes the proposed NS-RAD methodology, including the step-by-step operational workflow, the traceability between the problem and the method, and the differentiation from existing methods. Implementation details are reported in Section 4. We present empirical evaluation, ablation, and comparative analyses in Section 5. Section 6 concludes the paper and discusses directions for future research.

2. Literature Review

The quality of software requirements significantly influences the success or failure of software engineering projects. Two pervasive issues in requirements engineering are ambiguity, where requirements can be interpreted in multiple ways, and conflicts, where two or more requirements contradict one another. Both issues lead to misunderstandings, increased rework, cost overruns, and project delays, motivating a substantial body of research to detect, analyze, and resolve ambiguity and conflict in requirement documents.

2.1. Requirements conflict detection

Requirements conflicts happen when two or more requirement statements are inconsistent or mutually conflicting, such that no consistent and implementable software design can be derived. Stakeholder disagreements, overlapping or competing functionalities, or incompatible constraints on quality attributes like performance, security, or availability can lead to such conflicts. Poor requirements quality is well recognized as having a major dimension of conflicting requirements, which are known to undermine consistency, correctness, and completeness in SRS documents [6, 7].

The early work on conflict detection was largely formal and model-based. These approaches model requirements through structured or formal models such as goal models, feature models,

or constraint-based representations and apply predefined consistency rules to identify contradictions and incompatibility. For example, model-based conflict detection techniques systematically analyze requirement structures to detect violations of logical or semantic constraints, which enables early identification and resolution of conflicts if formal artifacts are available [8]. Guo et al. [9] proposed *Finer Semantic Analysis-based Requirements Conflict Detector (FSARC)* that translates natural-language requirements to semantic tuples of events, agents, operations, and constraints and uses heuristic rules over these tuples to find conflicts. Such techniques yield strong logical guarantees but require a lot of upfront modeling effort and do not scale to the heterogeneous, informally written SRS documents common in industrial practice.

Behavioral and temporal aspects of requirements are modeled using logical formalisms like fuzzy branching temporal logic [10] and linear temporal logic [11] based on formal logic techniques. These models can attain high accuracy in controlled environments by capturing the requirements' dependencies over time. But they demand that requirements be formally codified, creating complexity and limiting them to large-scale or informally written natural-language SRS documents. Similarly, ontology-based approaches [12] are aimed at providing a semantic representation of the ideas in the domain and their is-a relationships. Such systems detect conflicts by looking for violations of the semantic constraints of the ontology. These approaches provide a certain degree of conceptual depth, but they usually need detailed domain-specific ontologies, and they might have scalability problems in the case of multiple application domains.

To address these limitations, more recent approaches work directly on natural-language requirements, using distributional semantics and ML. S3CDA and its variant UnSupCDA first generate candidate requirement pairs based on sentence embeddings and similarity measures and then verify conflicts by checking shared entities, actions, and semantic cues [4]. Such approaches improve scalability and applicability to real-world requirement documents but heavily depend on hand-tuned similarity thresholds and often treat semantic similarity and linguistic features as largely independent signals, resulting in limited robustness across domains and requirement styles [4, 13]. In addition to automated detection, requirements negotiation and resolution techniques [14] manage conflicts through stakeholder and preference reconciliation. Recent dual-encoder transfer-learning methods, such as Transferable Software Requirement Conflicts Detection Framework based on SBERT and SimCSE (TSRCDF-SS), combine Sentence-Bidirectional Encoder Representations from Transformers (SBERT) and Simple Contrastive Sentence Embedding (SimCSE) representations and improve cross-domain conflict detection but remain primarily neural and provide limited symbolic evidence for analysts [15]. Table 1 summarizes the principal existing approaches and their reported characteristics.

Across the conflict detection literature, three weaknesses recur. First, operating thresholds are commonly selected for a particular dataset and are not updated by a documented learning rule. Second, similarity and dense-representation methods offer limited evidence about the linguistic or logical source of a predicted conflict. Third, analyst decisions are rarely incorporated through a governed, auditable adaptation mechanism. These limitations define the research gap taken forward after the literature review.

2.2. Requirements ambiguity detection

The main reason for ambiguity in software requirements is that specifications are written in natural language, which is

Table 1
Comparison of existing requirements conflict detection approaches

Approach	Technique	Explainability	Scalability	Feedback
Formal, Model-Based	Formal models, rules	High	Low	No
FSARC	Semantic tuples, heuristic	Medium	Medium	No
Semantic Similarity (S3CDA)	Sentence embeddings, cosine	Low	High	No
UnSupCDA	Unsupervised similarity pipeline	Low	High	No
Hybrid NLP + ML	Linguistic + ML features	Medium	High	No
Negotiation-Based Methods	Stakeholder negotiation	High	Low	Yes (manual)
TSRCDF-SS	SBERT + SimCSE dual encoders	Low-Medium	High	No

inherently ambiguous in lexical, syntactic, and semantic terms. Such ambiguity creates opportunities for various interpretations of the same requirement, which can result in inconsistent understanding between stakeholders and increase the likelihood of defects propagating into later development stages.

A large body of work exists on computational approaches for automatically detecting ambiguous requirements. To overcome the manual inspection and scalability for large specifications, ML and deep learning classifiers have been proposed to classify the ambiguity patterns in requirement sentences [16]. Along with learning-based approaches, fuzzy logic techniques such as Mamdani fuzzy inference systems [17] have been used for modeling linguistic vagueness and flagging possible ambiguous requirements.

Several studies examine semi-automated approaches that combine natural language processing (NLP) with soft computing or heuristic rules to detect and resolve ambiguous requirements, particularly in large-scale or industrial settings where manual review is impractical [18]. In parallel, prevention-oriented frameworks such as the Software Requirement Ambiguity Avoidance Framework [19] focus on improving requirements at the source.

Rule-based tools such as Automated Requirements Measurement (ARM) [20] and REquirements Template Analyzer (RETA) [21] use keyword lists, syntactic patterns, and shallow parsing to implement well-known ambiguity taxonomies and to identify constructs related to optionality, subjectivity, and under-specification. These tools provide transparent and interpretable diagnostics, but empirical evaluations show that they often have high false-positive rates (FPRs) and limited robustness to domain-specific language, limiting their effectiveness in industrial settings.

More recent work has focused on hybrid and learning-based methods for detecting ambiguity. Dictionary-based and ontology-supported approaches have been proposed to enhance the coverage of domain-specific ambiguous expressions, especially for non-English requirements. Rule-based detection along with statistical or ML classifiers in hybrid frameworks has proved better precision and scalability than rule-based methods alone [13]. The shared tasks of NLP4RE [22] also benchmarked ambiguity detection techniques on several datasets, where trade-offs between recall and precision are still persistent and the importance of explainability for ambiguity detection was emphasized.

Very recent work investigates the use of Large Language Models (LLMs) and deep learning for the detection of ambiguity. LLM-based studies show promise in explainable detection but also pose concerns about controllability, prompt sensitivity, and reproducibility in safety-critical settings [6]. Data-augmented neural approaches, such as the GloVe-BiLSTM framework, have reported a strong performance in classifying ambiguity on Fault-Prone SRS datasets, but they are focused on ambiguity rather than on pairwise conflict detection or feedback governance [23]. Table 2 summarizes these existing method families and their reported limitations.

Together, ambiguity detection research suffers from a lasting trade-off between coverage, precision, and interpretability. Rule-based methods reveal the evidence of triggering but often over-flag domain-specific language; learned methods improve contextual discrimination but require labeled data and can obscure the rationale for a decision. The existing studies also lack details as to how the analyst feedback should be used to update the decision policies over time. This gap motivates approaches to explicit evidence, contextual filtering, and governed adaptation.

Table 2
Comparison of SRS ambiguity detection methods

Approach	Core method and strength	Reported limitation
Rule-based tools (ARM, RETA)	Keyword lists and shallow parsing; transparent but brittle.	High false-positive rates and limited contextual or domain sensitivity.
Fuzzy/semi-automated approaches	Soft-computing rules and analyst clarification for large projects.	Depend on rule design and analyst clarification; limited standardization across domains.
Fine-tuned transformer classifiers	BERT/ROBERTa classifiers for sentence-level ambiguity.	Require labeled data and may provide limited evidence for individual decisions.
LLM-based ambiguity detection	Prompt-based detection and explanation with promising coverage.	Prompt sensitivity, non-determinism, and reproducibility concerns.
GloVe-BiLSTM with augmentation	Word embeddings, BiLSTM, and back-translation for ambiguity classes.	Ambiguity-only scope; performance depends on augmented data and domain transfer.

2.3. Neuro-symbolic and feedback-driven approaches

Together with individual ambiguity or conflict detectors, an increasing number of studies on neuro-symbolic artificial intelligence advocate the combination of neural learning with symbolic reasoning and explicit feedback processes to deal with uncertainty, contradictions, and domain drift in complex systems. Neuro-symbolic approaches combine the ability of neural models to generalize with the interpretability and ability to enforce constraints of symbolic representations and have been used for description-logic reasoning, cyber-physical system monitoring, and medical decision support [24–26]. Applications of neuro-symbolic and feedback-driven techniques in requirements engineering remain limited: existing tools are still dominated by either rule-based NLP or predominantly neural pipelines, and few studies specify how analyst feedback changes decision policies in a reproducible and auditable manner.

3. Neuro-Symbolic Requirements Anomaly Detection (NS-RAD)

This section presents NS-RAD as an executable methodology rather than only as conceptual architecture. The framework contains four layers, shown in Figure 1, and is operationalized through the step-by-step workflow in Algorithm 1.

3.1. Operational workflow of NS-RAD

Algorithm 1 defines the proposed method from raw requirement ingestion to analyst feedback. The inputs are natural-language SRS documents, use cases, user stories, and optional

metadata. Numerical decision thresholds are not entered by system operators: they are initialized from a held-out labeled calibration partition and subsequently recomputed from adjudicated feedback. The outputs are conflict reports, ambiguity reports, calibrated confidence scores, evidence traces, and a versioned feedback log used for adaptation.

3.2. Requirements ingestion and normalization

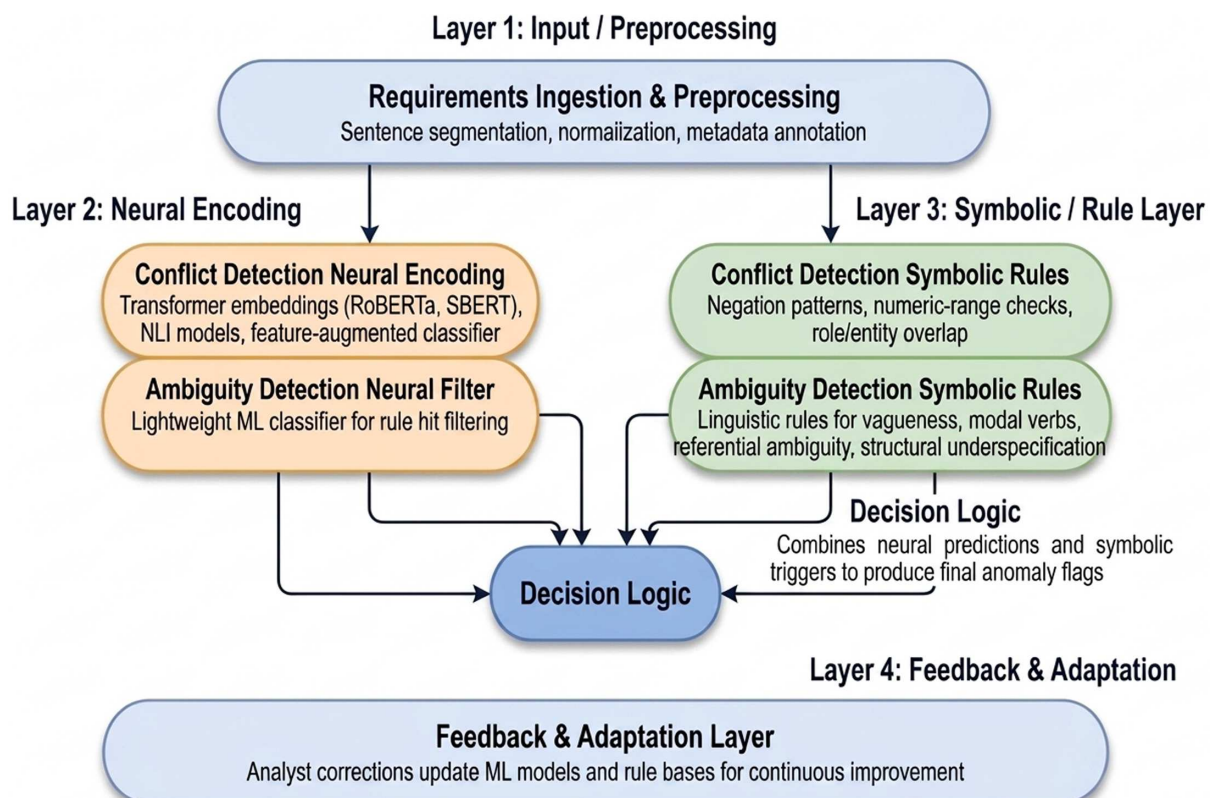
Layer 1 ingests heterogeneous requirements artifacts and normalizes them into sentence-level requirement units. To enable downstream layers to apply different sensitivity levels, each unit is enriched with contextual metadata such as features, priority, version, and lifecycle maturity. The preprocessing step includes domain-specific tokenization, acronym expansion, and template recognition to ensure consistency across specification patterns.

3.3. Multi-view semantic and linguistic encoding

Neuro-symbolic representations are constructed in Layer 2 of NS-RAD by integrating complementary semantic and linguistic views, explicitly recognizing that semantic similarity is not binary but exists on a continuum with varying risk levels:

- 1) **Neural semantic signals:** A distilled sentence transformer (SBERT) encodes graded semantic similarity for efficient candidate generation, and NLI models estimate entailment, neutrality, and contradiction probabilities to capture fine-grained logical relations. Such signals will help the system

Figure 1
NS-RAD architecture



- to disambiguate between high similarity, partial overlap, and semantically risky similarity.
- 2) **Symbolic linguistic features:** Develop interpretable linguistic cues for recognizing structural mismatches that might indicate hidden conflicts or ambiguities in the presence of surface similarity. These include part-of-speech tags, dependency relations, named entities, negation markers, quantifiers, modal verbs, and numeric expressions.
 - 3) **Similarity-risk stratification:** Sentence pairs are mapped into three semantic regimes. With the combined neural and symbolic evidence, these are low-risk similarity (clear equivalence or entailment), high-risk similarity (lexically or semantically close but logically divergent), and low similarity.

Algorithm 1: NS-RAD requirement anomaly detection algorithm

PHASE A – INPUT PREPARATION

- 01 Ingest SRS documents and metadata.
- 02 Segment artifacts into sentence-level requirement units.
- 03 Normalize SRS text.
- 04 Generate semantic embeddings.
- 05 Extract symbolic features.

PHASE B – CONFLICT DETECTION

- 06 Generate candidate conflict pairs.
- 07 Construct conflict features.
- 08 Classify each candidate pair as conflict, neutral, or requires-check.

PHASE C – AMBIGUITY DETECTION

- 09 Apply ambiguity rules.
- 10 Filter ambiguity triggers using a lightweight ML layer.

PHASE D – CALIBRATION, FEEDBACK, AND ADAPTATION

- 11 Calibrate detector scores and estimate maturity-specific lower and upper decision thresholds.
 - 12 Apply the three-way decision rule and generate structured reports.
 - 13 Record analyst labels in the anomaly log and test the automatic feedback-batch trigger.
 - 14 Refit calibrators, recompute and validate thresholds, version the accepted update, and update models or rules when scheduled.
-

3.4. Dual anomaly analyzers

Layer 3 implements two parallel, maturity-aware anomaly analysis pipelines for conflict and ambiguity detection. Each pipeline operates over the multi-view neuro-symbolic representations produced in Layer 2 and is designed to balance scalability, precision, and interpretability in industrial SRS settings.

1) Conflict detection pipeline

Conflict detection is realized as a two-stage process:

- a. Candidate generation: To avoid exhaustive $O(n^2)$ pair-wise comparisons, a lightweight retrieval stage identifies potentially conflicting requirement pairs using cosine similarity over RoBERTa sentence embeddings, shared named entities, and lexical feature overlap.
- b. Conflict classification: Candidate pairs are evaluated by a supervised neural classifier that integrates complementary semantic and symbolic signals. The input feature vector consists of a concatenation of contextual sentence embeddings, contradiction probabilities based on NLI, negation and polarity indicators, numeric-range and comparator features, semantic similarity scores, and role and entity overlap statistics.

The classifier is a feed-forward network with a configurable number of hidden layers and dropout regularization. It is trained with a weighted cross-entropy loss to account for the class imbalance. We employ early stopping on the validation performance. Each prediction is provided with a calibrated confidence score and in a machine-readable conflict report format.

2) Ambiguity detection pipeline

Ambiguity detection relies on a hybrid neuro-symbolic approach that combines explicit linguistic rules with a lightweight statistical refinement layer. The rule-based analysis is built upon a curated taxonomy of ambiguity patterns encoded with lexical resources, part-of-speech tags, dependency relations, and structural heuristics. The covered categories are vague or subjective adjectives, unbounded quantifiers, modal verbs, referential ambiguity, and structural under-specification. This rule layer provides explainable, transparent triggers consistent with the established requirements for engineering ambiguity taxonomies. In the statistical filtering and scoring phase, a lightweight ML filter is used to take advantage of contextual and domain-sensitive cues to reduce over-triggered rules. Ambiguities detected are marked as high confidence requires-check flags.

3) Maturity-aware analysis and outputs

Both pipelines apply the maturity-specific lower and upper decision thresholds defined in Section 3.5. Draft requirements use the most sensitive operating point, evolving requirements use an intermediate point, and baselined requirements use the strictest point rather than imposed through hand-selected offsets. Scores below the lower threshold are treated as neutral, scores inside the data-derived uncertainty band are routed to requires-check, and scores above the upper threshold are reported as high-confidence anomalies. Layer 3 outputs structured conflict reports with evidence traces and ambiguity highlights annotated by type, severity, and maturity context.

3.5. Automatic threshold calibration, feedback, and adaptation

Layer 4 separates human adjudication from numerical threshold computation. The anomaly log stores the detector name, item or pair identifier, maturity state, raw score, calibrated probability, evidence trace, predicted state, analyst label, severity, comment, and timestamp. Analysts supply confirmations or corrections; the log is a labeled calibration source, not a threshold-tuning interface.

Decision rule. For detector d and maturity state m , a logistic score calibrator maps the raw model score to an anomaly probability p . NS-RAD maintains a lower threshold $l(d,m)$ and an upper threshold $h(d,m)$. The status is then defined as per the rule below: if $p < l$, then it's "neutral" else; if $l \leq p < h$, then it is "requires-check" else; and if $p \geq h$, it is an "anomaly."

Automatic initialization. In each sample, candidate cutoffs are the unique calibrated probabilities. The conflict cutoff is the candidate that maximizes Matthews correlation coefficient (MCC), whereas the ambiguity cutoff maximizes F2 to preserve the recall-oriented operating policy. For each bootstrap replicate, the draft, evolving, and baselined cutoffs are projected to the nondecreasing order where $\text{draft} \leq \text{evolving} \leq \text{baseline}$ by weighted isotonic regression. If a maturity stratum lacks both labels, the global detector cutoff is used for that stratum. The 10th and 90th percentiles of the bootstrap-optimal cutoffs become τ_L and τ_H , so the requires-check band is data-derived rather than hand-selected.

Feedback-triggered update. A detector becomes eligible for recalibration after 50 new adjudicated cases have accumulated and both positive and negative labels are present. The update pool combines the original calibration partition with at most the 500 most recent adjudicated cases, giving recent domain evidence influence without discarding the original operating baseline. The logistic calibrator is refitted, and the bootstrap procedure is repeated. A candidate threshold version is deployed only when the lower bound of the paired 95% bootstrap confidence interval for the change in the detector objective is non-negative; otherwise, the previous version is retained automatically.

3.6. Problem-method traceability

Table 3 explicitly links the research problem stated in the introduction to the operational steps of NS-RAD. This traceability clarifies how each stage of the proposed method contributes to accurate, scalable, interpretable, and workflow-compatible anomaly detection.

4. Implementation Details

The implementation of NS-RAD shows its neuro-symbolic design and interpretable, scalable, and adaptive anomaly

detection. Both the conflict detection and ambiguity detection modules exploit complementary neural and symbolic representations and combine them in a unified pipeline.

4.1. Datasets

To ensure that every quantitative result can be independently reconstructed, the evaluation uses only publicly available SRS benchmarks (Table 4): OpenCoss, WorldVista, unmanned aerial vehicle (UAV), and the NLP4RE ambiguity corpus. No unpublished author-curated dataset contributes to model fitting, threshold calibration, or the metrics reported in Section 5. The three conflict corpora span transportation, healthcare, and aerospace and include logical negation, parameter mismatch, scope variation, and mutually exclusive operations.

- 1) The OpenCoss dataset is extracted from the Open Platform for Evolutionary Certification of Safety-Critical Systems, containing 6786 transportation-related requirements (railway, avionics) with repetitive syntactic patterns and paraphrases.
- 2) The WorldVista dataset contains 10878 healthcare-related requirements featuring domain-specific but relatively simple language, emphasizing semantic generalization across nontechnical contexts.
- 3) The UAV dataset contains 4329 functional requirements for UAV control systems. Conflicts involve parameter mismatches, logical contradictions, and mutually exclusive operations, providing structurally regular yet semantically rich examples.

All conflict datasets were preprocessed using sentence-level normalization, manual label verification, duplicate removal, and cosine-similarity-based neutral-pair sampling to preserve semantic diversity. Table 4 reports the source-corpus sizes; after duplicate removal, the UAV analysis set contained 4,090 pairs, which is the count used in Tables 5 and 6. Requirement pairs are labeled as conflict or neutral. Model fitting and score calibration use training and held-out calibration partitions, and the resulting threshold version is frozen before final evaluation.

The ambiguity detection module was evaluated using the referential ambiguity subtask from the NLP4RE shared task (ReqEval benchmark). The corpus contains approximately 200–212 single-sentence requirements from seven domains (spacecraft

Table 3
NS-RAD method traceability

Problem	NS-RAD contribution
Heterogeneous and noisy SRS text	Steps 1–3 create comparable, traceable sentence-level requirement units.
Expensive pairwise conflict search	Step 6 reduces unnecessary comparisons through candidate-pair generation.
Similarity alone cannot detect contradictions	Steps 4–8 combine embeddings, NLI, negation, and numeric checks.
Rule-only ambiguity tools overproduce false positives	Steps 9–10 combine linguistic triggers with contextual ML filtering.
Analysts need actionable outputs	Steps 11–12 calibrate scores and provide severity, confidence, evidence spans, and recommended actions.
Domain terminology changes over time	Steps 13–14 use adjudicated feedback to version threshold, rule, lexicon, and classifier updates.
Hand-tuned decision thresholds	Steps 11, 13, and 14 derive maturity-specific decision bands through calibration, stratified bootstrap, automatic validation, and rollback; analysts do not choose cutoff values.

Table 4
Summary of datasets used for conflict and ambiguity detection

Dataset	Domain	#Pairs/sen- tences	Conflict type	Neutral type	Public	Synthetic
OpenCoss	Transportation: railway, avionics, repetitive syntactic patterns	6786	Logical, paraphrase conflicts	Paraphrases	Yes	No
WorldVista	Healthcare: patient info systems, domain-specific language	10878	Parameter mismatches, contradictions	Diverse neutral pairs	Yes	No
UAV	Aerospace: UAV functional requirements, Easy Approach to Requirements Syntax (EARS) style	4329	Logical contradictions, mutually exclusive ops	EARS paraphrases	Yes	No
NLP4RE (Ambiguity)	Mixed: seven domains, referential ambiguity sentences	212	Referential ambiguity	Clear pronouns	Yes	No

Table 5
Corpus-level statistics and NS-RAD predicted scores for public duplicate/related text-pair datasets

Dataset	#Pairs	Avg Tok T1	Avg Tok T2	Avg Char T1	Avg Char T2	Vocab	TTR	Avg Jac.	Avg Cos.	Avg conflict	Avg ambiguity
UAV	4090	43.1	262.8	256.5	1501.7	25109	0.02	0.056	0.039	0.12	0.08
OpenCoss	6786	28.2	28.2	176.9	179.2	250	0.007	0.339	0.247	0.25	0.15
WorldVista	10878	20.8	19.5	129.0	124.0	884	0.002	0.140	0.055	0.21	0.12

Table 6
Extended conflict and ambiguity statistics with percentiles and risk-level counts for NS-RAD

		Conflict						Ambiguity					
Dataset	Pairs	Median	90 th	95 th	Low risk	Med risk	High risk	Median	90 th	95 th	F1	Pre	Rec
UAV	4090	0.10	0.22	0.30	3520	450	120	0.07	0.18	0.25	0.66	0.58	0.78
OpenCoss	6786	0.20	0.45	0.55	4800	1600	386	0.12	0.28	0.36	0.87	0.97	0.78
WorldVista	10878	0.18	0.40	0.50	8500	1800	578	0.11	0.25	0.33	0.92	0.99	0.76

control, diagnostic systems, digital home, digital library, weather services, archive file manipulation, and railway control). Five experts annotated pronouns as ambiguous or clear. A pronoun is labeled ambiguous if at least one annotator perceived ambiguity and clear only if all annotators agreed, producing a conservative, recall-oriented gold standard. Markup tags were removed for evaluation to simulate realistic SRS text.

4.2. Conflict detection module

The conflict detection module uses transformer-based semantic representations and engineered linguistic and logical features to model the conflict risk between requirement pairs in a graded and context-aware manner.

1) **Neural semantic representations:** We extract contextual sentence embeddings with RoBERTa to encode high-level semantic content and support scalable candidate-pair generation. We retain the embedding similarity as a continuous signal to distinguish the different levels of semantic overlap and the corresponding risk of conflict rather than binary classification on semantic similarity.

- 2) **Risk-aware symbolic and augmented feature signals:** We extend the feature set with a range of engineered features that can capture logical incompatibilities that can occur even when the two sentences are semantically similar, including NLI contradiction scores, cosine similarity between embeddings, explicit negation cues, and named entity or role overlap statistics. These features enable the classifier to distinguish benign similarity from high-risk contradictions involving negation, numeric constraints, or incompatible roles.
- 3) **Classification architecture:** The concatenated feature vector is fed to a feed-forward neural network trained using a weighted cross-entropy loss to take into account class imbalance. We use validation-based early stopping for generalization and stability on heterogeneous SRS datasets. Instead of a hard binary decision, the classifier produces calibrated confidence scores that represent the estimated risk of conflict.

The present module differs from similarity-only or rule-based conflict detection approaches in that it explicitly models semantic similarity as a graded, risk-aware signal and jointly learns its interaction with symbolic linguistic cues and logical inference features.

4.3. Ambiguity detection module

The ambiguity detection module operates on tokenized and syntactically parsed requirement sentences, producing sentence-level ambiguity flags with associated categories and confidence levels. The rule-based detection engine encodes ambiguity patterns as executable rules derived from established requirements engineering taxonomies. The implementation combines curated lexical resources with part-of-speech tags, dependency relations, and shallow structural patterns to detect vague or subjective expressions, modal verbs, unbounded quantifiers, referential ambiguity, and structurally underspecified constructions. The span that triggers the rule, and the typed ambiguity signal it produces, enables fine-grained diagnostics. The statistical filtering layer controls over-triggering, a light supervised filter. This component uses local contextual features and domain-specific cues to reduce systematic false positives while keeping the interpretability of rule activations. The filter is trained on a small, annotated subset and can be updated incrementally as new feedback becomes available.

The implementation splits high-confidence ambiguity flags from lower-confidence requires-check indicators, which enables analyst triage instead of binary decisions. This design strikes a balance between recall and precision without resorting to large-scale supervised learning, making the module useful in industrial contexts with sparse annotated ambiguity data and a need for explainability.

4.4. Neuro-symbolic integration and adaptation

At the implementation level, NS-RAD combines neuro-symbolic methods with explicit feature fusion and traceable decision logic instead of opaque end-to-end learning. We treat neural representations (sentence embeddings and NLI-derived scores) as continuous semantic signals and symbolic linguistic features as discrete, interpretable indicators. These heterogeneous signals are fused with structured feature vectors and rule trigger outputs, maintaining predictive power and explainability.

The implementation uses logistic score calibration (Section 3.5). Accepted thresholds are serialized with the calibrator parameters and feedback identifiers, enabling exact rollback and audit of every operating-policy change. Classifier fine-tuning and ambiguity-rule exceptions are maintained as separate versioned updates so that a change in model parameters cannot be confused with a change in the score-to-decision mapping.

For the experiments reported here, the automatic calibration procedure was run once on held-out calibration scores, and the resulting threshold version was locked before test evaluation. Because the public benchmarks do not contain time-ordered analyst adjudications, the feedback-triggered recalibration cycle was not executed in the reported experiments; its longitudinal performance is reserved for industrial evaluation.

5. Results and Discussions

This section presents and discusses the experimental results for the proposed NS-RAD framework. The performance is separately evaluated for conflict detection and ambiguity detection, followed by additional metrics, ablation-style component analysis, and a comparative discussion against representative and recent methods. Comparisons with external studies are explicitly marked as literature-reported when the baselines were not rerun in the same experimental setup.

5.1. Evaluation metrics

We assessed the performance of both ambiguity and conflict detection using standard classification metrics derived from confusion-matrix entries: accuracy, precision, recall, F1-score, specificity, FPR, false-negative rate, balanced accuracy, and MCC. Because conflict cases are sparse in many SRS corpora, precision, recall, F1-score, balanced accuracy, and MCC are emphasized over raw accuracy. All score calibrators and lower/upper decision thresholds were fitted on held-out calibration data and frozen before final evaluation; test labels were not used for threshold selection.

5.2. Conflict detection results

Conflict detection was evaluated on the three public conflict datasets. Table 5 reports corpus-level statistics, including token and character counts (Avg Tok T1/T2, Avg Char T1/T2), vocabulary size (Vocab), type-token ratio (TTR), Jaccard similarity (Avg Jac.), cosine similarity (Avg Cos.), and NS-RAD predicted anomaly scores (Avg Conflict, Avg Ambiguity). Average conflict scores ranged from 0.12 to 0.25, consistent with the sparse occurrence of true conflicts.

Extended statistics (Table 6) show that the median, 90th, and 95th percentiles capture subsets of pairs with elevated conflict scores. Risk stratification reveals that OpenCoss contains the largest fraction of medium- and high-risk pairs, reflecting syntactic repetition and semantic overlap, whereas UAV and WorldVista are dominated by low-risk pairs with smaller high-risk tails.

The final RoBERTa-based conflict detector achieved an F1-score of 0.8545 (precision 0.9792, recall 0.7581) on held-out validation data, demonstrating improved generalization and calibration relative to the initial BERT model, which overfit the training data. The operating threshold and requires-check band were produced by the automatic calibration procedure in Section 3.5 and frozen before evaluation. Compared with S3CDA-style baselines and recent dual-encoder methods, NS-RAD combines transformer embeddings, NLI, engineered linguistic features, and traceable evidence rather than relying on similarity or dense representations alone [4, 15].

The requires-check output is not based on a manually selected margin around a point threshold. It is the interval between the 10th and 90th percentiles of bootstrap-optimal thresholds, so uncertain scores are routed for review according to calibration uncertainty rather than operator preference.

5.3. Ambiguity detection results

We evaluated ambiguity detection using the referential ambiguity subtask of the NLP4RE benchmark. Table 7 shows the main precision, recall, and F1-score results, as well as error analysis from an interpretational point of view. The module achieved an F1-score of 0.665 with a high recall (0.78) and a moderate precision (0.58). This is in line with the conservative design of the module to maximize the coverage of ambiguous cases while keeping the flagged spans interpretable. False positives usually arise when ambiguous terms from general language are used with precise meanings in a particular domain or when multiword phrases only partially match rule patterns. False negatives come from subtler pragmatic or scenario-dependent ambiguities. The detector identifies spans of subjective adjectives, unbounded quantifiers, modal verbs with unclear obligations, and pronouns with

Table 7
Ambiguity detection measures and error analysis on the NLP4RE benchmark

Measure	Value and interpretation
Precision	0.58. Moderate positive predictive value; prioritize high-confidence ambiguity flags.
Recall	0.78. Most ambiguous requirements are retrieved, supporting conservative early QA.
F1-score	0.665. The rule plus ML-filter strategy provides useful triage and room for domain tuning.

ambiguous antecedents. These outputs provide actionable guidance yet avoid excessive flagging.

Extended NS-RAD metrics (Table 6) complement corpus-level averages reported in Table 5 by providing median, 90th, and 95th percentiles of conflict and ambiguity scores and pair counts by risk levels. Across datasets, median ambiguity scores remain below 0.20, but the 90th and 95th percentiles reveal a high-risk tail, underscoring NS-RAD's ability to identify potentially problematic pairs. Medium- and high-risk ambiguity instances are particularly concentrated in healthcare- and aerospace-related datasets.

5.4. Comparative discussion

Conflict detection consistently achieves higher precision and F1-scores than ambiguity detection, reflecting more abundant labeled data and the clearer structural and semantic cues present in requirements expressing contradictions. Ambiguity detection, by contrast, is inherently recall-oriented due to the subtlety and sparsity of referential and lexical ambiguities in annotated corpora, consistent with observations in prior studies on NLP4RE and rule-based ambiguity identification. Error profiles differ between the tasks: conflict detection false negatives primarily stem from implicit domain knowledge, conditional clauses, or complex numeric constraints, whereas ambiguity detection false positives typically involve contextually precise but linguistically ambiguous expressions, such as technical terms with dual interpretations.

The automatic three-way decision policy is also relevant to interpretation of the results. Scores near the operating boundary are routed to requires-check instead of being forced into anomaly or neutral labels. This behavior is governed by bootstrap

uncertainty in the calibrated threshold and is applied consistently across maturity states. Comparative claims should nevertheless remain method-level rather than leaderboard claims because external studies use different datasets, splits, and operating policies.

In summary, NS-RAD demonstrates cross-domain performance, interpretable reasoning, and calibrated risk assessment relative to similarity-based conflict detectors, semantic-tuple models, neural dual-encoder models, LLM-based ambiguity detectors, and rule- or template-driven ambiguity classifiers [4, 6, 15, 23]. On public datasets and held-out partitions, NS-RAD prioritizes high-risk anomalies while maintaining review state for uncertain cases. The combination of conflict and ambiguity detection provides complementary quality assurance: conflict scores can direct inspection of high-risk pairs of requirements, and ambiguity highlights can direct analysts' attention to language that may cause confusion.

Because the external baselines were not all rerun on identical splits, Table 8 is interpreted as a literature-positioning comparison rather than a strict leaderboard. The NS-RAD figures are reported from the present experimental setup, whereas external values are literature-reported.

5.5. Threats to validity

Several threats to validity should be considered when interpreting the results. First, the public corpora differ in domain, pair-construction procedure, and label distribution, which limits direct cross-dataset comparison. Second, some comparative results are literature-reported rather than reproduced on identical splits, so they are used for positioning rather than statistical ranking. Third, the NLP4RE benchmark emphasizes referential

Table 8
Comparative positioning against representative and recent conflict and ambiguity detection methods

Method	Reported strength	How NS-RAD differs
S3CDA/UnSupCDA [4]	Scalable conflict candidate identification using similarity and semantic filtering.	Adds NLI, numeric checks, negation, role overlap, ambiguity detection, and feedback.
TSRCDF-SS [15]	Dual SBERT + SimCSE; reports 4.9%–12.1% macro/weighted-F1 improvement.	Adds symbolic evidence traces and automatically recalibrated maturity-aware decision bands.
Rule-based NLP/ChatGPT [3]	Useful preliminary ambiguity identification through rules/prompts.	Combines rules with ML filtering and governed feedback.
LLM-based ambiguity explanation [6]	Promising explanation-augmented ambiguity detection.	Avoids prompt-only decisions and records feature/rule evidence.
GloVe-BiLSTM with augmentation [23]	Reports P = 92.04%, R = 91.52%, F1 = 91.65%, Acc = 92.06% on Fault-Prone SRS.	Focuses on ambiguity only; NS-RAD also handles pairwise conflicts.
NS-RAD	Conflict F1 = 0.8545 on validation; ambiguity F1 = 0.665 on NLP4RE.	Unified, interpretable, maturity-aware, feedback-driven, and automatically calibrated.

ambiguity, whereas lexical, pragmatic, and domain-specific ambiguity require broader validation. Fourth, the automatic threshold procedure is evaluated here only at initialization because the benchmarks do not contain a chronological stream of analyst feedback. Maturity-specific feedback recalibration, the 50-label trigger, and the deployment guard therefore require longitudinal industrial validation; when a maturity stratum lacks both labels, the method falls back to the global detector threshold.

6. Conclusions and Future Work

In this paper, we propose an AI-enabled framework that combines a transformer-based, feature-augmented conflict detector with a linguistically grounded ambiguity module to detect conflicts and ambiguities in natural-language software requirements. The conflict detector depends on RoBERTa embeddings, NLI signals, and engineered features. The ambiguity module encodes lexical, syntactic, and contextual patterns, using statistical filtering and interpretable evidence. A fully specified calibration layer produces maturity-aware neutral, requires-check, and anomaly decision bands from held-out labels and updates them from adjudicated feedback without operator-selected cut-off values. Evaluation on public datasets shows strong conflict detection, useful ambiguity triage, and actionable outputs for requirement analysts and QA teams. NS-RAD draws attention to conflicting pairs and ambiguous statements at an early stage, thus reducing rework and increasing specification quality.

Future work includes extending evaluation to additional domains, languages, and styles of requirements, rerunning recent neural and LLM baselines on a unified benchmark, and expanding ablation experiments with statistical significance testing. A priority is longitudinal industrial evaluation of the feedback-triggered threshold updates, including sensitivity to the update-batch size, recent-feedback window, maturity-stratum sparsity, and rollback guard. The conflict and ambiguity modules may also be integrated into end-to-end neural architectures that jointly predict multiple anomaly types while retaining rules and annotations as governed weak supervision.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in UAV at <https://www.kaggle.com/datasets/ziya07/uav-autonomous-navigation-dataset>, in NLP4RE at <https://data.mendeley.com/datasets/zs9vkk7sb7/1>, in OpenCOSS at <https://github.com/open-compass/opencompass>, and in WordIVista at <https://worldvista.org/>.

Author Contribution Statement

Anis Zarrad: Conceptualization, Methodology, Formal analysis, Data curation, Supervision, Project administration. **Jaber Jemai:** Conceptualization, Validation, Investigation, Resources,

Writing – review & editing, Visualization, Supervision. **Danyal Abbasi:** Software, Data curation, Writing – original draft, Visualization. **Yassine Daadaa:** Validation, Resources.

References

- [1] Cruz, Rosado da., M. A., & Cruz, E. F. (2025). Machine learning techniques for requirements engineering: A comprehensive literature review. *Software*, 4(3), 14. <https://doi.org/10.3390/software4030014>
- [2] Necula, S. C., Dumitriu, F., & Greavu-Şerban, V. (2024). A systematic literature review on using natural language processing in software requirements engineering. *Electronics*, 13(11), 2055. <https://doi.org/10.3390/electronics13112055>
- [3] Fantechi, A., Gnesi, S., & Semini, L. (2023). Rule-based NLP vs ChatGPT in ambiguity detection, a preliminary study. In *CEUR Workshop Proceedings* (p. 3378).
- [4] Malik, G., Cevik, M., Başar, A., & Parikh, D. (2025). Supervised semantic similarity-based conflict detection algorithm: S3CDA. In *2025 IEEE International Conference on Collaborative Advances in Software and Computing*, 1–10. <https://doi.org/10.1109/CASCON66301.2025.00079>
- [5] Khalil, I., Ahmad, I., Rasheed, U., Butt, W. H., & Anwaar, Z. (2025). Detecting cross-domain ambiguity in requirements through natural language processing, a systematic literature review. In *2025 6th International Conference on Advancements in Computational Sciences*, 1–7. <https://doi.org/10.1109/ICACS64902.2025.10937875>
- [6] Bashir, S., Ferrari, A., Khan, A., Strandberg, P. E., Haider, Z., Saadatmand, M., & Bohlin, M. (2025). Requirements ambiguity detection and explanation with LLMs: An industrial study. In *2025 IEEE International Conference on Software Maintenance and Evolution*, 620–631. <https://doi.org/10.1109/ICSME64153.2025.00063>
- [7] Farooq, A., & Faisal, M. (2023). Assessing similarity between software requirements: A semantic approach. *International Journal of Information Engineering and Electronic Business*, 15(2), 38. <https://doi.org/10.5815/ijieeb.2023.02.05>
- [8] Wahab, A., Wang, J., Shojaei, A., & Ma, J. (2023). A model-based smart contracts system via blockchain technology to reduce delays and conflicts in construction management processes. *Engineering, Construction and Architectural Management*, 30(10), 5052–5072.
- [9] Guo, W., Zhang, L., & Lian, X. (2021). Automatically detecting the conflicts between software requirements based on finer semantic analysis. *arXiv Preprint: 2103.02255*
- [10] Viana, T., Zisman, A., & Bandara, A. K. (2017). Identifying conflicting requirements in systems of systems. In *2017 IEEE 25th International Requirements Engineering Conference*, 436–441. <https://doi.org/10.1109/RE.2017.48>
- [11] Elekes, M., Molnár, V., & Micskei, Z. (2023). Assessing the specification of modelling language semantics: A study on UML PSSM. *Software Quality Journal*, 31(2), 575–617. <https://doi.org/10.1007/s11219-023-09617-5>
- [12] Sangaroonsilp, P., Choetkiertikul, M., Dam, H. K., & Ghose, A. (2023). An empirical study of automated privacy requirements classification in issue reports. *Automated Software Engineering*, 30(2), 20. <https://doi.org/10.1007/s10515-023-00387-9>
- [13] Jamasb, B., Khayami, S. R., Akbari, R., & Taheri, R. (2025). An automated framework for prioritizing software

- requirements. *Electronics*, 14(6), 1220. <https://doi.org/10.3390/electronics14061220>
- [14] Terpstra, E. (2015). A systematic literature review of requirements negotiation methods from 2010 till 2015. *Student Undergraduate Research E-journal*, 1. <https://doi.org/10.25609/sure.v1.1051>
- [15] Wang, Y., Jiang, T., Bai, J., Zou, Z., Xue, T., Zhang, N., & Luan, J. (2026). A transfer learning approach for automatic conflicts detection in software requirement sentence pairs based on dual encoders. *PLOS ONE*, 21(3), e0344174. <https://doi.org/10.1371/journal.pone.0344174>
- [16] Alharbi, S. (2022). Ambiguity detection in requirements classification task using fine-tuned transformation technique. In *CS & IT Conference Proceedings*, 12, 47–55. <https://doi.org/10.5121/csit.2022.122105>
- [17] Mamman, H., Balogun, A. O., Basri, S., Capretz, L. F., Adeyemo, V. E., Imam, A. A., & Kumar, G. (2023). Software requirement risk prediction using enhanced fuzzy induction models. *Electronics*, 12(18), 3805. <https://doi.org/10.3390/electronics12183805>
- [18] Asadabadi, M. R., Saberi, M., Zwikael, O., & Chang, E. (2020). Ambiguous requirements: A semi-automated approach to identify and clarify ambiguity in large-scale projects. *Computers & Industrial Engineering*, 149, 106828. <https://doi.org/10.1016/j.cie.2020.106828>
- [19] Gupta, A. K., & Deraman, A. (2019). A framework for software requirement ambiguity avoidance. *International Journal of Electrical and Computer Engineering*, 9(6), 5436–5445. <https://doi.org/10.11591/ijece.v9i6.pp5436-5445>
- [20] Durrani, U. K., Akpınar, M., Adak, M. F., Kabakus, A. T., Öztürk, M. M., & Saleh, M. (2024). A decade of progress: A systematic literature review on the integration of AI in software engineering phases and activities (2013-2023). *IEEE Access*, 12, 171185–171204. <https://doi.org/10.1109/ACCESS.2024.3488904>
- [21] Bajceta, A., Leon, M., Afzal, W., Lindberg, P., & Bohlin, M. (2022). Using NLP tools to detect ambiguities in system requirements - a comparison study. In *5th Workshop on Natural Language Processing for Requirements Engineering*.
- [22] Abualhaija, S., Aydemir, F. B., Dalpiaz, F., Dell'Anna, D., Ferrari, A., Franch, X., & Fucci, D. (2024). Replication in requirements engineering: the NLP for RE case. *ACM Transactions on Software Engineering and Methodology*, 33(6), 1–33. <https://doi.org/10.1145/3658669>
- [23] Abdeahad, Y., Kabootari, M., Kheirkhah, Y., Kheirkhah, E., Sheibani, R., & Moattar, M. H. (2026). Accurate and robust ambiguity detection in software requirements documents using a GloVe-BiLSTM deep learning framework with data augmentation. *Scientific Reports*. <https://doi.org/10.1038/s41598-026-48700-2>
- [24] Liang, B., Wang, Y., & Tong, C. (2025). AI reasoning in deep learning era: From symbolic AI to neural-symbolic AI. *Mathematics*, 13(11), 1707. <https://doi.org/10.3390/math13111707>
- [25] Garcez, A. D. A., & Lamb, L. C. (2023). Neurosymbolic AI: the 3rd wave. *Artificial Intelligence Review*, 56(11), 12387–12406. <https://doi.org/10.1007/s10462-023-10448-w>
- [26] Zarrad, A., Armstrong, T., & Jemai, J. (2026). A hybrid approach to software testing efficiency: Stacked ensembles and deep Q-Learning for test case prioritization and ranking. *Computers, Materials and Continua*, 86(3). <https://doi.org/10.32604/cmc.2025.072768>

How to Cite: Zarrad, A., Jemai, J., Daadaa, Y., & Abbasi, D. (2026). NS-RAD: A Neuro-Symbolic Framework for Conflict and Ambiguity Detection in Natural-Language Requirements. *Artificial Intelligence and Applications*. <https://doi.org/10.47852/bonviewAIA620210263>