

# AI-DRIVEN EXPLICIT AND IMPLICIT INFORMATION COLLECTION IN REQUIREMENTS ENGINEERING: CLASSIFICATION AND KNOWLEDGE REPRESENTATION

AHMAD MURAD <sup>1</sup>, KHALIL JAMOUS <sup>1\*</sup>, IHAB ATIEH <sup>1</sup>, AMJAD HUDAIB <sup>1</sup>, NADIM OBEID <sup>1</sup>, MARWAN AL-TAWIL <sup>1</sup>

<sup>1</sup> Department of Computer Science, King Abdullah II School of Information Technology, University of Jordan, Amman, Jordan.

Email: AHM9240269@ju.edu.jo , \* KLY9230267@ju.edu.jo , AYH9240030@ju.edu.jo , nadim@ju.edu.jo , ahudaib@ju.edu.jo , M.AITawil@ju.edu.jo

## ABSTRACT

The evolving complexity of software systems demands smarter Requirements Engineering (RE) methods that can uncover both explicit and implicit user needs. Traditional RE techniques often miss latent contextual information, leading to incomplete or suboptimal system specifications. This paper introduces an AI-enhanced framework that integrates Natural Language Processing (NLP), Machine Learning (ML), and Knowledge Graphs (KGs) to intelligently extract, and classify requirements. Explicit requirements are derived from structured documentation using BERT and TF-IDF, while implicit requirements are inferred from behavioral patterns and unstructured data. Class imbalance is mitigated using SMOTE and ADASYN techniques, and deep learning via BiLSTM captures bidirectional semantic dependencies. The PROMISE dataset is used to train and evaluate the model, with results benchmarked using accuracy, F1-score, and ontology completeness. The resulting knowledge graph enables contextual traceability. Our framework demonstrates improved requirement visibility, and semantic enrichment, moving RE toward a more intelligent, context-aware discipline.

**Keywords:** *Requirements Engineering RE, Explicit and Implicit Requirements, Knowledge Graphs KG, Natural Language Processing NLP, Machine Learning ML, AI-Driven Requirements Analysis.*

## 1. INTRODUCTION

In modern software development, the success of a system often hinges on the completeness, accuracy, and adaptability of its requirements. Traditional Requirements Engineering (RE) practices primarily focus on eliciting explicit requirements, those that are clearly articulated by stakeholders or written into specifications. However, in many complex and user-driven environments, a large portion of the requirements exist implicitly, embedded in stakeholder behavior, historical documents, system logs, and even organizational routines (1). Requirements Engineering (RE) serves as the critical foundation for building systems that fulfill stakeholder expectations. However, traditional RE processes often emphasize explicitly stated requirements, those gathered through interviews, questionnaires, and documented specifications. While neglecting implicit requirements embedded in stakeholder behavior, historical data, or

contextual clues. This oversight results in information asymmetries that undermine software quality and system adaptability (2). However, despite the importance of both explicit and implicit knowledge in RE, existing practices still lack robust mechanisms for systematically identifying, classifying, and representing implicit requirements at scale. Current methods often treat requirement elicitation as a static process, leaving a gap in automated discovery, semantic understanding, and traceability, particularly when requirements are unstructured, ambiguous, or evolve over time. This represents a significant research problem that the field has yet to fully address.

To tackle this duality, recent research has proposed AI-driven approaches that leverage Natural Language Processing (NLP), Machine Learning (ML), and Knowledge Graphs (KGs) to support the intelligent discovery and interpretation of both explicit and implicit

requirements (3). These methods bring automation and reasoning capabilities to RE, enabling dynamic requirement identification, traceability, and prioritization. With the rise of Artificial Intelligence (AI) and Knowledge Graphs (KGs), there is a unique opportunity to bridge this gap. AI technologies such as natural language processing (NLP) and machine learning (ML) are now capable of extracting and interpreting both explicit and implicit information at scale. Simultaneously, knowledge graphs allow for the structured, semantic representation of complex requirement interrelations, aiding prioritization, traceability, and dynamic updates (4).

Knowledge Graphs, in particular, have emerged as powerful structures for representing complex relationships between requirement entities. They facilitate semantic reasoning and can support dynamic priority redefinition based on the contextual interdependencies of system components (5). Combined with deep learning, KGs can extract, complete, and reason over both structured and unstructured knowledge from diverse requirement sources (6). Moreover, Knowledge Graphs act as a connective tissue between fragmented data formats, transforming siloed insights into a semantic network of interconnected knowledge (7). They enable both explicit representation of requirements and implicit inference mechanisms using ontology-driven reasoning (8).

Recent studies have also demonstrated the use of generative AI and large language models to interpret vague or ambiguous requirement statements and convert them into actionable engineering tasks (9). These models, when integrated with knowledge graph backends, show promise in supporting requirement evolution, real-time conflict detection, and informed trade-off analysis (10). Moreover, the need for context-aware and adaptive RE systems has led to increased research on goal-driven and ontology-based models (11), which align well with KG-centric approaches. This synergy also supports traceability across lifecycle stages, from conceptual design to system implementation by enabling automated linkage of requirement rationale and domain concepts (12). Despite these advances, challenges persist in extracting latent requirements from unstructured data, modeling semantic conflicts among interdependent goals, and ensuring that requirement prioritization reflects evolving user and business needs (13).

Addressing these gaps calls for a framework that unifies AI-based extraction, KG reasoning, and dynamic priority redefinition.

Therefore, the objective of this study is to develop and evaluate an AI-enhanced framework that integrates NLP, ML, and KG-based reasoning to address the research gap in identifying and classifying explicit and implicit requirements. The proposed approach applies BERT-based embeddings, TF-IDF extraction, class balancing techniques (SMOTE, ADASYN, SMOTE-ENN), and deep learning through BiLSTM to improve requirement classification accuracy. The classified requirements are then transformed into a domain-specific Knowledge Graph to support semantic representation and reasoning.

By combining statistical, semantic, and graph-based methods, this study aims to contribute a unified, intelligent RE framework that enhances requirement visibility, contextual understanding, and traceability, addressing a critical need in both industry and academic RE research.

## 2. BACKGROUND

The automation of Requirements Engineering (RE) has evolved significantly in recent years, driven by advancements in Natural Language Processing (NLP), Machine Learning (ML), Deep Learning (DL), and semantic knowledge representation. However, despite this progress, challenges remain in the simultaneous extraction of explicit requirements, inference of implicit requirements, and semantic structuring of requirement knowledge. This section reviews prior work across these domains, identifies key limitations, and highlights the research gap addressed by the proposed framework.

### 2.1 Requirements Engineering: Explicit and Implicit Knowledge

Requirements Engineering (RE) serves as the foundation for aligning system functionality with stakeholder intent. Traditionally, RE has focused on gathering explicit requirements, those formally articulated through interviews, documentation, or specifications. However, this approach often overlooks implicit requirements, which are unspoken assumptions, user expectations, or contextual dependencies embedded in communication patterns, behaviors, and historical

data (1). Recent efforts have focused on formalizing implicit knowledge extraction, recognizing that unexpressed needs can lead to feature gaps or stakeholder dissatisfaction. Researchers such as (3) have proposed frameworks that integrate implicit cues using AI-enhanced models for model-driven requirement extraction.

## 2.2 AI and NLP in Requirements Engineering

The integration of Natural Language Processing (NLP) and Machine Learning (ML) into RE has enabled automated parsing of large-scale documentation and feedback, improving both scalability and accuracy (5). NLP-based RE tools can extract intent, entities, and relations from natural text using pretrained models such as BERT or domain-specific transformers. The integration of Artificial Intelligence (AI), especially Natural Language Processing (NLP) and Machine Learning (ML) has facilitated automated approaches to capture both explicit and implicit requirements. (3) introduced a model-driven RE framework that uses NLP and semantic analysis to detect hidden intents in stakeholder documents. Similarly, (14) provide a comprehensive review of how ML is transforming requirements elicitation, particularly through pattern recognition and data-driven refinement. Generative approaches using LLMs (Large Language Models) are emerging as powerful tools for interpreting ambiguous stakeholder inputs, transforming them into structured requirement artifacts (9). These approaches also support summarization, prioritization, and dialog-based clarification.

## 2.3 Knowledge Graphs in RE

Knowledge Graphs (KGs) have emerged as robust tools for organizing, connecting, and reasoning over extracted requirements. By structuring domain-specific knowledge into nodes and edges, KGs enable traceability, impact analysis, and consistency checks (6). They also allow inference of implicit links between requirements and help identify redundancies or conflicts. Research by (10). has shown how KGs combined with NLP enhance the adaptability of RE artifacts. Similarly, (11). demonstrated KG-based curriculum design that aligns educational goals (a form of requirements) to evolving learner data, a method easily translatable to software RE. The use of Knowledge Graphs (KGs) offers a semantic and scalable method for representing interconnected requirement data. KGs encode

both formal (explicit) and informal (implicit) relationships, enabling advanced reasoning and prioritization strategies. For example, (15) developed a knowledge representation system using NLP to map requirements documents to a KG structure.

## 2.4 Priority Redefinition through AI Reasoning

While conventional RE often fixes priorities early, real world systems evolve dynamically. Priorities should shift based on new dependencies, emerging constraints, or user behavior patterns. AI techniques, particularly graph reasoning and deep learning-based prioritization, have been proposed for these challenges (12). A reinforcement learning-based KG reasoning framework was introduced by (5), enabling intelligent inference of updated priorities based on task dependencies and feedback loops.

## 2.5 Human-AI Collaboration and Ethical Concerns

Collaborative RE systems that include AI agents raise questions of explainability, bias detection, and trust. (13) discussed knowledge graph-enhanced human-AI teaming systems that ensure transparent requirement decisions, especially in regulatory contexts. This reinforces the need for interpretable AI pipelines in RE workflows.

## 2.6 Deep Learning enhanced KG models

Deep Learning enhanced KG models are emerging to link knowledge mining with neural feature extraction. (16) use NLP and transfer learning to model autonomous knowledge mining and implicit requirement discovery from construction documents. This aligns with the work of (17), who developed ontology testing techniques powered by knowledge graph embeddings. The synergy between Large Language Models (LLMs) and KGs is also gaining attention. (18) explore this interplay, emphasizing how LLMs can inject deep contextual reasoning into KGs for requirements interpretation. Similarly, (19) discuss challenges in aligning symbolic reasoning with neural-based systems for better transparency and scalability.

## 2.7 Hybrid systems

Hybrid systems combining logical reasoning and ML have also been proposed. (20) demonstrated that hybrid reasoning frameworks

can assist goal modeling in RE, enabling systems to incorporate implicit domain knowledge and logical constraints. Likewise, (21) proposed leveraging KGs to create a machine learning engineering body of knowledge, stressing the need to represent tacit engineering practices formally. Further extending this, systems like those described by (22) classify NLP techniques for RE based on the origin of knowledge (explicit vs. implicit), suggesting classification pipelines that continuously refine requirements from various sources. (23) conducted a systematic review of KGs in software engineering, underlining their role in implicit knowledge embedding and semantic enrichment. Some systems also embed commonsense reasoning into NLP models. (24) combined text mining with commonsense knowledge to locate implicit requirements in stakeholder narratives. In alignment, (25) proposed a model integrating AI planning with NLP to balance explicit procedural steps and tacit knowledge during system specification.

### 2.8 Summary of Research Gaps

Across the literature, several persistent limitations are identified:

- **Fragmented Approaches:** Existing solutions address isolated components, classification, implicit detection, or KG construction, rather than offering an integrated framework.
- **Difficulty Handling Implicit Requirements:** Few studies provide robust mechanisms for detecting implicit or context-dependent requirements from unstructured data.
- **Weak Multi-Class NFR Performance:** Many models achieve high binary accuracy but significantly lower results for full multi-class classification.
- **Limited Semantic Integration:** Knowledge Graphs are underutilized as reasoning tools that can enhance classification and traceability.

These gaps highlight the need for a unified, AI-driven approach that integrates advanced NLP models (e.g., BERT), class imbalance solutions (SMOTE/SMOTE-ENN), deep learning architectures (BiLSTM), and semantic KG representation.

### 2.8 Contribution of the Proposed Work

The proposed framework addresses the above limitations by:

- Combining NLP, ML, and DL techniques for end-to-end classification of explicit and implicit requirements.
- Integrating class imbalance handling to improve NFR subtype recognition.
- Employing BERT + BiLSTM for superior contextual understanding.
- Constructing a domain-specific Knowledge Graph for semantic enrichment, traceability, and reasoning.
- Providing a unified pipeline that bridges the gap between extraction, classification, and semantic representation.

### 3. RELATED WORK

There have been a few researches done on the topic in the past 5 years. In the (Table 1), we have summarized the most related papers that has a very close relation to the work done in our research. The summary table presents a comparative analysis of various machine learning and deep learning approaches for classifying non-functional requirements (NFRs) in software engineering, highlighting their models, methodologies, and performance metrics. (26) employed traditional machine learning algorithms like Logistic Regression, SVM, Multinomial Naive Bayes, and KNN with TF-IDF and Chi-Squared feature extraction, achieving a solid 91% accuracy and 0.91 F1-score for binary classification, though performance dropped for general classification (0.78 F1-score). (27) utilized a Linear Support Vector Classifier with TF-IDF vectorization, slightly improving accuracy to 91.5% with high precision for NFR classification. In contrast, (28) leveraged a CNN with a Flower Pollination Optimizer and SMOTE, demonstrating superior performance with accuracies ranging from 94.48% to 97.13% on balanced datasets and 87.45% to 98% on unbalanced ones, reflecting robustness across data conditions. (29) applied deep learning models (CNN, LSTM, GRU), achieving a high 96% accuracy with strong precision and F1-scores, underscoring the efficacy of deep learning for NFR tasks. Lastly, (30) used a Bidirectional GRU, achieving 90% accuracy for binary and 85% for multiclass classification, with slightly lower precision and F1-scores for multiclass tasks. Collectively, the table illustrates a trend where deep learning models (e.g., CNN, GRU) outperform traditional machine learning approaches in accuracy and robustness, particularly when enhanced by advanced optimization and balancing techniques, though

traditional methods remain competitive for specific binary classification tasks.

Table 1: Related Work Papers from 2020 till 2025

Paper	Model & Methodology	Accuracy	Precision	F1-Score
[26]	Logistic Regression (LR), Support Vector Machine (SVM), Multinomial Naive Bayes (MNB), k-Nearest Neighbors (KNN) with TF-IDF and Chi-Squared feature extraction	91% (binary classification)	Varied per model	0.91 (binary), 0.74 (NF classification), 0.78 (general classification)
[27]	Linear Support Vector Classifier with TF-IDF vectorization	91.5%	High precision for NF classification	91.5%
[28]	CNN with Flower Pollination Optimizer (FPO) and SMOTE for class balancing	94.48% - 97.13% (balanced dataset), 87.45% - 98% (unbalanced dataset)	High precision	High F1-score
[29]	Deep learning-based classification of non-functional requirements using CNN, LSTM, and GRU	96%	High precision	High F1-score
[30]	Bidirectional Gated Recurrent Unit (BiGRU)	90% (binary classification); 85% (multiclass).	89% (binary); 83% (multiclass).	90% (binary); 84% (multiclass).

#### 4. METHODOLOGY

This section describes the proposed AI-driven framework for classifying software requirements and constructing a knowledge graph representation. The methodology is designed to be reproducible, with a clearly defined sequence of stages: data acquisition, preprocessing, feature extraction, class imbalance handling, model training and evaluation, and knowledge graph construction.

##### 4.1 Data Source

Requirements classification. The dataset contains textual requirement statements labeled as:

- Functional Requirements (FR)
- Non-Functional Requirements (NFR), further divided into subcategories such as performance, security, usability, reliability, maintainability, and others (see Table 3).

In our experiments, the PROMISE dataset is used for two main tasks:

- Binary classification: FR vs. NFR
- Full multi-class classification: FR and NFR subtypes (14 classes in total)

The dataset is split into training (80%) and testing (20%), stratified by class to preserve label distribution.

##### 4.2 System Pipeline

The system pipeline (see Figure 1) consists of the following sequential stages:

1. Data Import: Extraction of raw software requirement entries from the PROMISE dataset.
2. Preprocessing: Includes lowercasing, stop-word removal, and punctuation cleanup.
3. Tokenization & Embedding:
  - a. TF-IDF for keyword-based importance.
  - b. BERT for context-rich word embeddings.
4. Handling Minority Classes:
  - a. SMOTE and ADASYN for synthetic oversampling.
  - b. SMOTE-ENN for noise reduction in boundary samples.
5. Data Splitting: 80/20 ratio into training and testing sets.

6. Model Training: Using BiLSTM to capture sequential dependencies from both directions.
7. Evaluation: Metrics include precision, recall, F1-score, and accuracy.
8. Knowledge Graph Construction: Labeled requirements are semantically structured into a KG.

### 4.3 Tools & Frameworks

1. NLP Toolkit: HuggingFace Transformers (BERT), Scikit-learn (TF-IDF)
2. Imbalance Handling: imbalanced-learn (SMOTE, ADASYN, SMOTE-ENN)
3. Modeling: TensorFlow/Keras for BiLSTM
4. Semantic Modeling: RDFLib, neo4j for KG construction and Cypher querying

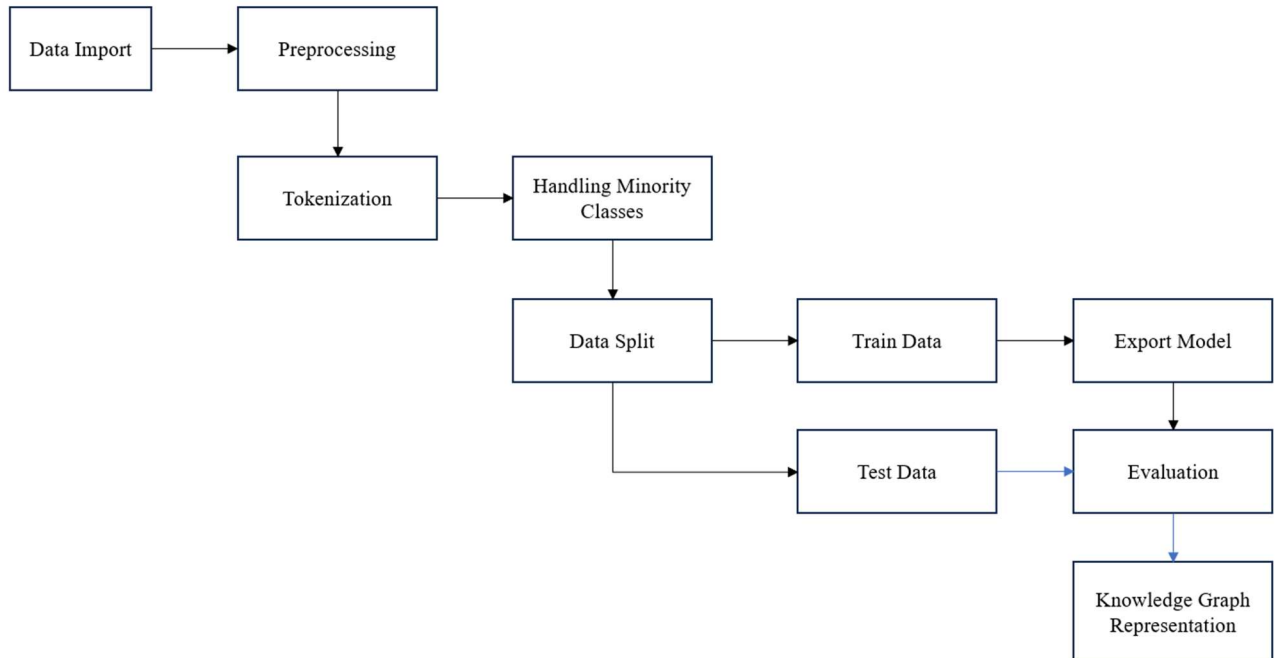


Figure 1: System Architecture Overview

## 5. OVERVIEW OF THE EXPERIMENTS

We developed eight distinct hybrid systems and evaluated each based-on accuracy, precision, and F1-score. These combinations were inspired by prior related studies but were enhanced and modified through custom Python implementations and libraries. All models were tested using the same dataset and executed on a consistent hardware environment with the following specifications:

- Processor: AMD Ryzen 7 5700G with Radeon Graphics, 3.80 GHz
- Installed RAM: 32.0 GB (31.4 GB usable)
- System Type: 64-bit operating system, x64-based processor

The experimental methodology incorporated a variety of algorithms and techniques for tokenization, handling class imbalance, and

selecting machine learning algorithms. Table 2, summarizes the different models and the specific algorithmic choices used in each.

### 5.1 experiments design

We conducted eight experiments aimed at classifying software requirements into two main tasks: (1) binary classification of requirements into functional (FR) and non-functional (NFR) categories, and (2) multi-class classification of NFRs into specific subcategories such as performance, security, usability, and reliability. Each experiment employed a unique combination of tokenization techniques (TF-IDF or BERT), class imbalance handling methods (SMOTE, ADASYN, SMOTE-ENN, or none), and machine learning algorithms (Random Forest, LightGBM, Multi-Layer Perceptron Classifier [MLPC], BiLSTM using TensorFlow or PyTorch, and Random Forest with K-Fold cross-validation).

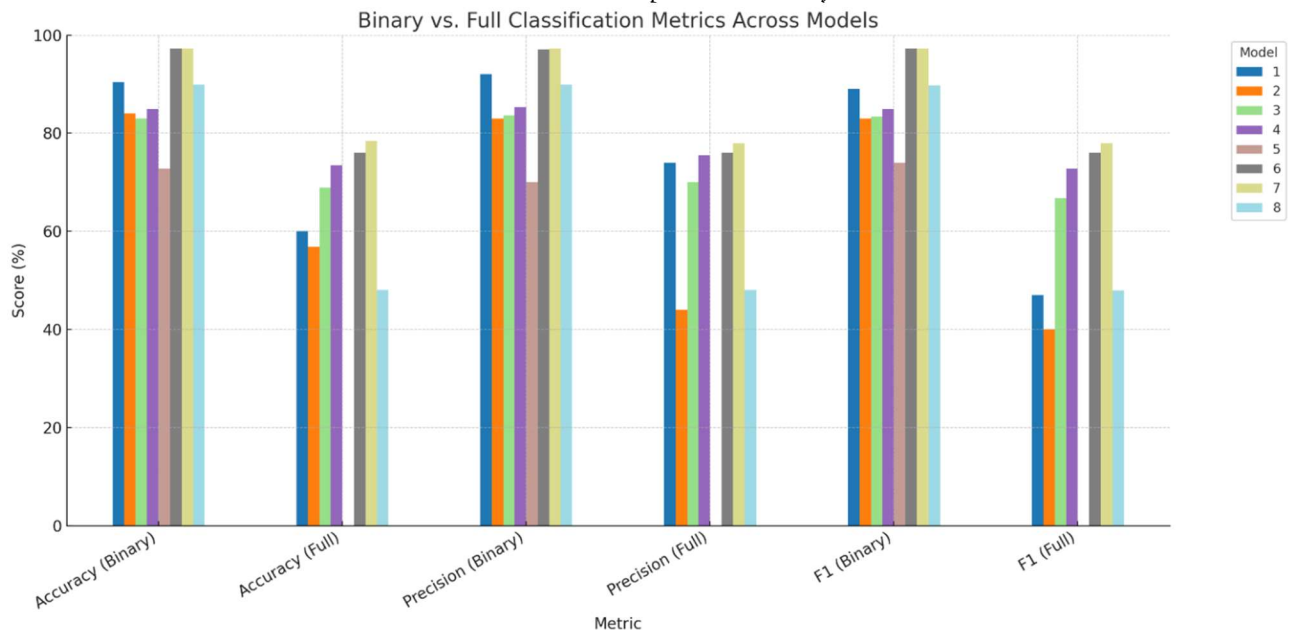
Although the dataset is not explicitly detailed, it is presumed to consist of software requirement statements and to exhibit class imbalance, particularly within NFR subtypes, justifying the application of data balancing techniques. Model performance was evaluated using standard metrics: accuracy, precision, and F1-score for both classification tasks. The experimental design integrates a variety of approaches, including traditional machine learning methods (e.g., Random Forest, LightGBM), deep learning

techniques (e.g., MLPC, BiLSTM), and advanced natural language processing strategies (e.g., BERT). The use of class balancing methods indicates a focus on mitigating the underrepresentation of certain NFR categories. The overarching objective was to determine the most effective model configuration for accurately automating both binary and multi-class classification tasks in the context of requirements engineering.

Table 2: Models Experiment Summary

Model	Tokenization Method	Minority class handling algorithm	Machine Learning algorithm	Classification		Accuracy		Precision		F1-Score
				Binary	Full	Binary	Full	Binary	Full	Binary
1	TF-IDF	SMOTE	Random Forest	Yes	Yes	90.4	60	92	74	89
2	TF-IDF	SMOTE	LGBM	Yes	Yes	84	56.8	83	44	83
3	BERT	ADASYN	Random Forest + K-Fold	Yes	Yes	83	68.9	83.6	70	83.3
4	BERT	SMOTE	MLPC + K-Fold	Yes	Yes	84.96	73.4	85.3	75.5	84.98
5	BERT	No	BiLSTM + Tensor Torch	Yes	No	72.8	N/A	70	N/A	74
6	BERT	SMOTE	BiLSTM + Tensor Torch	Yes	Yes	97.27	76.0	97.0	76	97.2
7	BERT	SMOTE-ENN	BiLSTM + Tensor Torch	Yes	Yes	97.27	78.4	97.27	78	97.2
8	BERT	SMOTE	Random Forest	Yes	Yes	89.9	48	89.9	48	89.8

Chart 1: Models Experiment Summary



### 5.2 Analysis of Binary Classification Performance (Functional vs. Non-Functional)

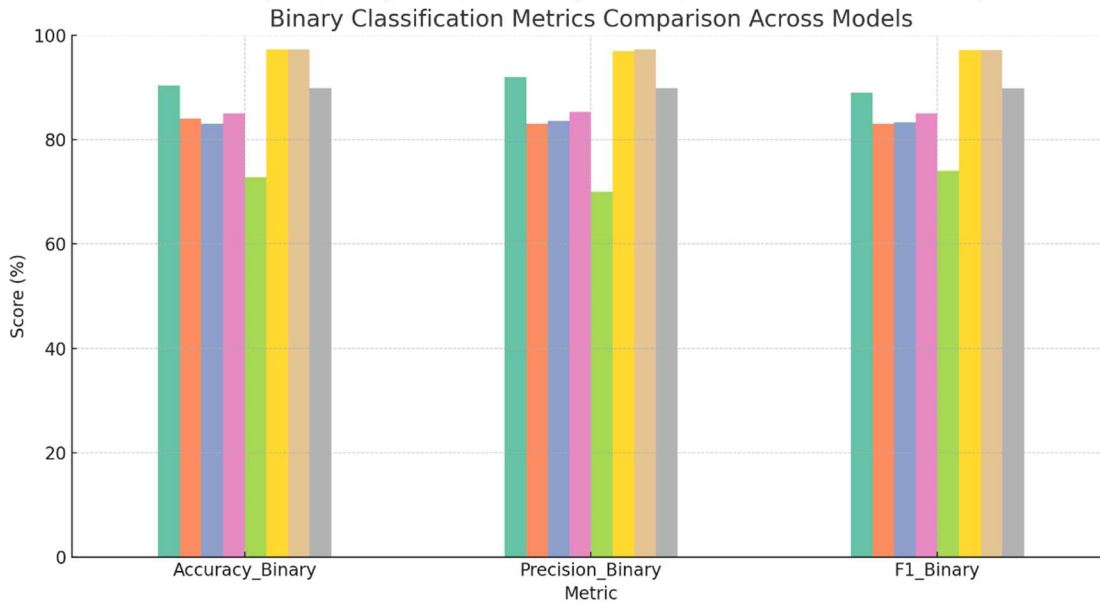
The binary classification task (FR vs. NFR) yielded a range of performance metrics across the eight experiments, with accuracy ranging from 72.8% (Model 5) to 97.27% (Models 6 and 7). Precision and F1-scores followed similar trends, indicating that some models were significantly more effective at distinguishing FRs from NFRs.

- **Top Performers (Models 6 and 7):** Both Models 6 and 7, which used BERT tokenization, SMOTE (or SMOTE-ENN), and BiLSTM with TensorFlow/Torch, achieved the highest accuracy (97.27%), precision (97.0–97.27%), and F1-score (97.2%). These results suggest that the combination of BERT’s contextual embeddings, effective minority class handling, and BiLSTM’s ability to model sequential dependencies in text was highly effective. The near-identical performance of Models 6 and 7 indicates that SMOTE-ENN (which combines oversampling with undersampling) provided marginal improvement over SMOTE alone, likely by reducing noise in the oversampled data.
- **Strong Performers (Models 1, 4, 8):** Model 1 (TF-IDF, SMOTE, Random Forest) achieved
- **iveness compared to SMOTE.**

a solid accuracy of 90.4%, with high precision (92%) and F1-score (89%). Model 8 (BERT, SMOTE, Random Forest) performed similarly, with an accuracy of 89.9%. Model 4 (BERT, SMOTE, MLPC with K-Fold) had a slightly lower accuracy (84.96%) but maintained strong precision (85.3%) and F1-score (84.98%). These models demonstrate that Random Forest and MLPC, when paired with appropriate tokenization and class balancing, can achieve robust performance, though they fall short of BiLSTM-based models.

- **Lower Performers (Models 2, 3, 5):** Model 5 (BERT, no minority class handling, BiLSTM) had the lowest accuracy (72.8%), precision (70%), and F1-score (74%). The absence of minority class handling likely exacerbated the impact of class imbalance, leading to poor performance despite using BERT and BiLSTM. Model 2 (TF-IDF, SMOTE, LightGBM) and Model 3 (BERT, ADASYN, Random Forest with K-Fold) also underperformed, with accuracies of 84% and 83%, respectively. LightGBM’s lower performance may reflect its sensitivity to feature quality, while Model 3’s use of ADASYN might have introduced noisy synthetic samples, reducing effect

Chart 2: Analysis of Binary Classification Performance (Functional vs. Non-Functional)



### 5.2.1 Key Insights

- BERT-based models (Models 3–8) generally outperformed TF-IDF-based models (Models 1–2) due to BERT’s ability to capture semantic and contextual nuances in requirement texts. For example, requirements like “The system shall be user-friendly” (NFR, usability) benefit from BERT’s understanding of context, whereas TF-IDF relies on keyword frequency, which may miss subtle distinctions.
- Minority class handling was critical. Model 5’s poor performance without SMOTE highlights the dataset’s imbalance, likely with fewer NFRs than FRs. SMOTE and SMOTE-ENN consistently improved performance across models.
- BiLSTM models (Models 5–7) excelled when paired with SMOTE, suggesting that sequential modeling of text is particularly effective for requirements classification.

### 5.3 Analysis of Full Classification Performance (Non-Functional Subtypes)

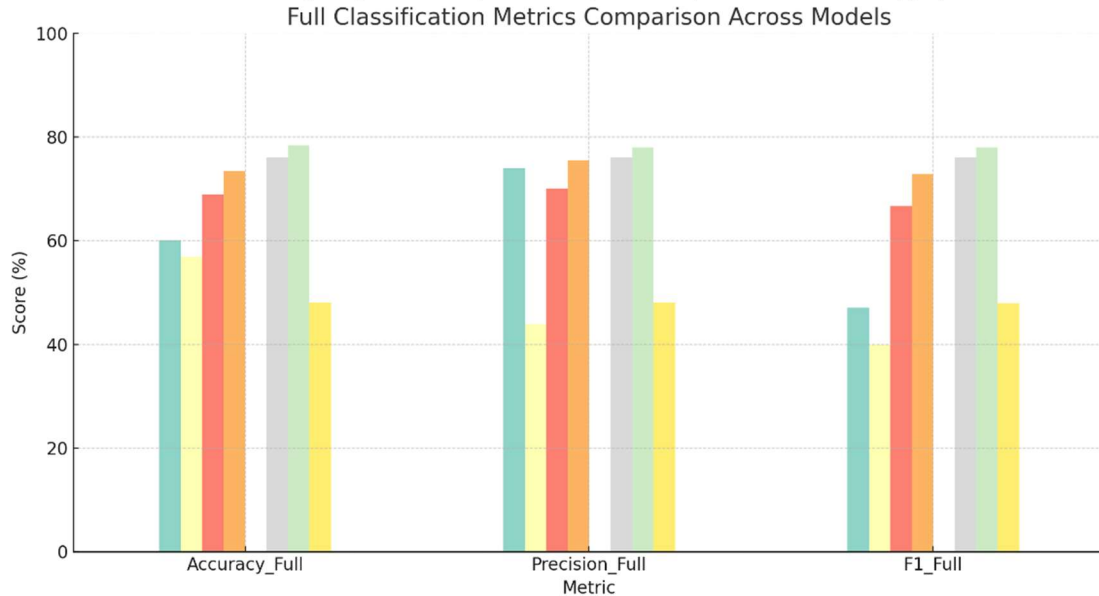
The full classification task, which involved categorizing NFRs into subtypes, was more challenging, with accuracies ranging from 48% (Model 8) to 78.4% (Model 7). Precision and F1-scores mirrored these trends, indicating significant variability in the models’ ability to distinguish NFR subtypes.

- Top Performer (Model 7): Model 7 (BERT, SMOTE-ENN, BiLSTM) achieved the highest full classification accuracy (78.4%), precision (78%), and F1-score (78%). This model’s success likely stems from its robust pipeline: BERT’s contextual embeddings captured nuanced differences between subtypes (e.g., “The system shall respond within 2 seconds” for performance vs. “The system shall be intuitive” for usability), SMOTE-ENN balanced the dataset while reducing noise, and BiLSTM modeled sequential patterns effectively. The 78.4% accuracy, while lower than binary

classification, is notable given the complexity of multi-class classification with potentially overlapping subtypes.

- Strong Performers (Models 4, 6): Model 6 (BERT, SMOTE, BiLSTM) achieved a full classification accuracy of 76%, closely trailing Model 7. The slight performance gap suggests that SMOTE-ENN’s noise reduction provided a marginal benefit. Model 4 (BERT, SMOTE, MLPC with K-Fold) also performed well, with an accuracy of 73.4%, precision of 75.5%, and F1-score of 72.8%. MLPC’s ability to model non-linear relationships, combined with K-Fold cross-validation, likely contributed to its robustness.
- Moderate Performers (Models 1, 3): Model 1 (TF-IDF, SMOTE, Random Forest) and Model 3 (BERT, ADASYN, Random Forest with K-Fold) had accuracies of 60% and 68.9%, respectively. Model 1’s lower performance may reflect TF-IDF’s limitations in capturing semantic differences between NFR subtypes, while Model 3’s use of ADASYN might have introduced synthetic samples that confused the Random Forest classifier.
- Lower Performers (Models 2, 8): Model 2 (TF-IDF, SMOTE, LightGBM) and Model 8 (BERT, SMOTE, Random Forest) performed poorly, with accuracies of 56.8% and 48%, respectively. Model 8’s particularly low performance is surprising given its use of BERT and SMOTE, suggesting that Random Forest struggled with the multi-class nature of full classification, possibly due to insufficient feature engineering or hyperparameter tuning. Model 2’s reliance on TF-IDF likely limited its ability to differentiate subtle NFR subtypes.
- Not Applicable (Model 5): Model 5 did not perform full classification, focusing only on binary classification. This limits its applicability for tasks requiring detailed NFR categorization.

Chart 3: Analysis of Full Classification Performance (Non-Functional Subtypes)



### 5.3.1 Key Insights

- Full classification was significantly harder than binary classification, likely due to the increased number of classes and potential overlap between NFR subtypes (e.g., usability and reliability requirements may share similar phrasing).
- BERT-based models consistently outperformed TF-IDF-based models, highlighting the importance of contextual embeddings for multi-class tasks.
- The best-performing models (6 and 7) used BiLSTM, suggesting that sequential modeling is critical for distinguishing NFR subtypes, which often rely on sentence structure and context.
- The low performance of Model 8 indicates that even advanced tokenization (BERT) cannot compensate for a suboptimal classifier (Random Forest) in complex multi-class settings.

### 5.4 Comparison Across Experiments

The (table 2) reveals clear trends and differences across the experiments:

- Tokenization Impact: BERT-based models (3–8) generally outperformed TF-IDF-based models (1–2) in both binary and full classification. For example, Model 6 (BERT, SMOTE, BiLSTM) achieved 97.27% binary accuracy compared to Model 1's 90.4% (TF-

IDF, SMOTE, Random Forest). This gap widened in full classification (76% vs. 60%), underscoring BERT's superiority for capturing semantic nuances.

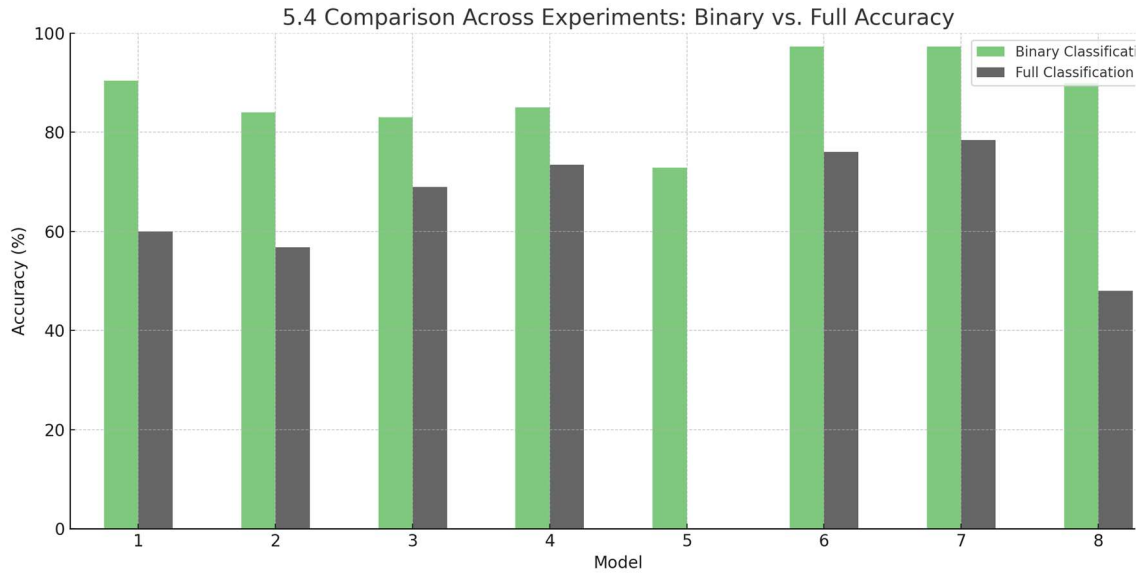
- Minority Class Handling: SMOTE and SMOTE-ENN were the most effective techniques, as seen in Models 6 and 7's high performance. ADASYN (Model 3) yielded moderate results, possibly due to noisy synthetic samples. The absence of minority class handling (Model 5) led to the lowest binary classification performance, confirming the dataset's imbalance.
- Machine Learning Algorithms: BiLSTM (Models 5–7) outperformed other algorithms when paired with BERT and SMOTE, achieving the highest metrics in both tasks. Random Forest (Models 1, 3, 8) and MLPC (Model 4) were competitive in binary classification but struggled in full classification, likely due to their limited ability to model complex text patterns. LightGBM (Model 2) was the least effective, possibly due to its reliance on feature quality from TF-IDF.
- K-Fold Cross-Validation: Models 3 and 4 used K-Fold cross-validation, which likely improved their robustness by reducing overfitting. However, their performance was still lower than Models 6 and 7, suggesting that model architecture (BiLSTM) and minority class handling (SMOTE-ENN)

were more critical than cross-validation alone.

- Consistency: Models 6 and 7 showed consistent performance across metrics, with

near-identical binary classification results and close full classification results. This stability suggests that BiLSTM-based models are robust to variations in minority class handling (SMOTE vs. SMOTE-ENN).

Chart 4: Comparison Across Experiments



### 5.5 Ontology development

We develop a domain-specific ontology that defines the key concepts, relationships, and properties relevant to software requirements (PROMISE\_exp dataset). This ontology serves as the schema for the Knowledge Graph, ensuring consistency and semantic clarity. Incorporate classes such as "Functional Requirement," "Non-Functional Requirement" and define relationships like "HAS\_REQUIREMENT," "BELONGS\_TO\_CLASSIFICATION".

Table 3: Classification Nodes With Full Names

Class name	Class code
Functional	F
Availability	A
Legal	L
Look-and-feel	LF
Functional Requirement	F
Maintainability	MN
Operability	O
Performance	PE

Scalability	SC
Security	SE
Usability	US
Fault Tolerance	FT
Portability	PO
Total	14

### 5.6 Knowledge graph representation

The preprocessed requirements mapped onto the ontology to construct the Knowledge Graph. Each requirement becomes a node, and relationships between requirements are represented as edges. the Knowledge Graph Snapshot (figure 2 and figure 3) illustrating the semantic structure of requirement relationships in the applied framework. Each node is a requirement, while edges define relations. This snapshot demonstrates how contextual dependencies are visually traceable and computable.

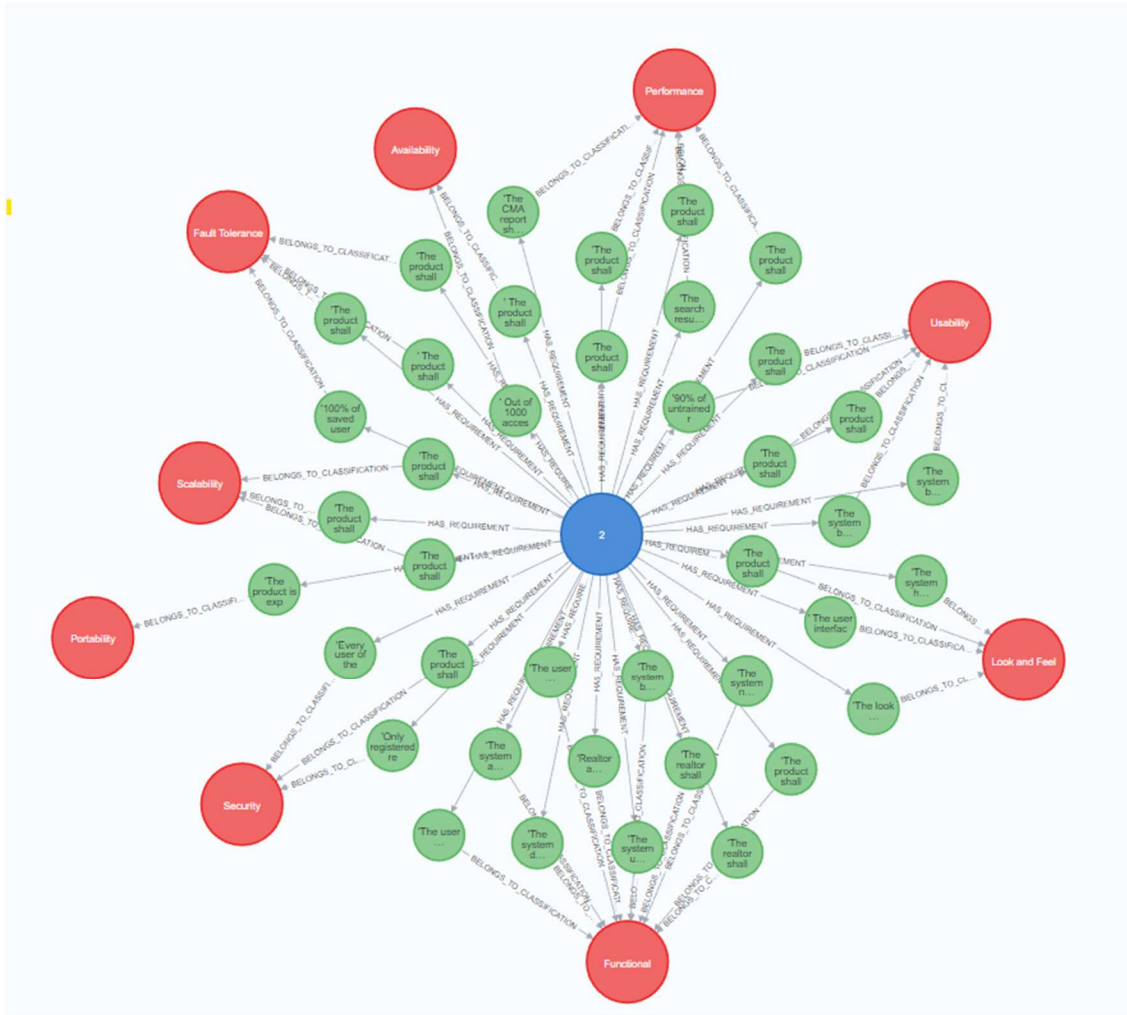


Figure 2: Knowledge Graph Snapshot Of Requirements And Relationships

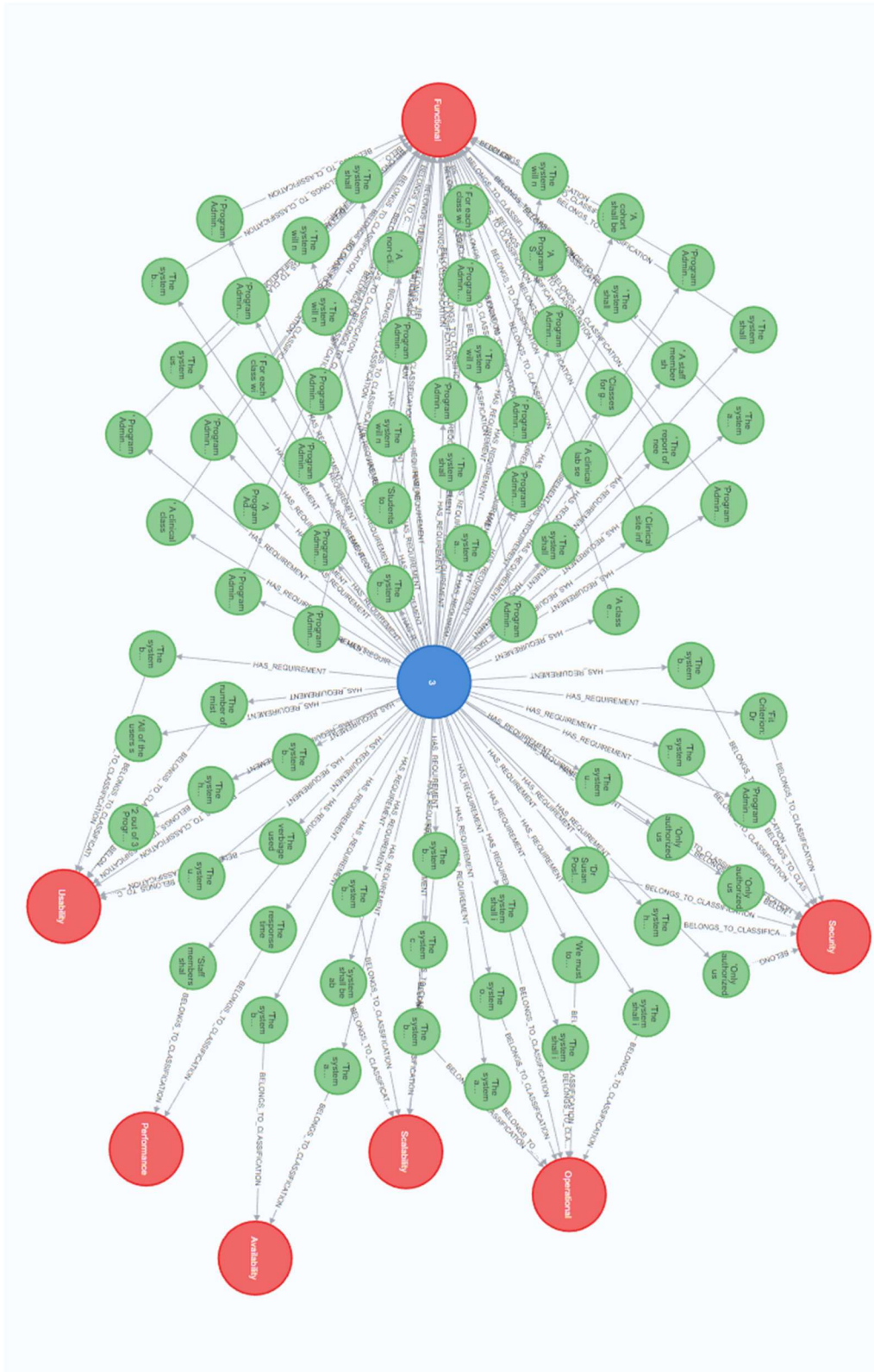


Figure 3: Knowledge Graph Snapshot Of Requirements And Relationships

## 6. CHALLENGES AND LIMITATIONS

Several challenges and limitations are evident from the results:

- **Class Imbalance:** The need for minority class handling indicates an imbalanced dataset, with fewer NFRs or specific NFR subtypes. Model 5's poor performance without SMOTE underscores this issue, as the model likely overfit to the majority class (FRs).
- **Ambiguity in Requirements:** NFR subtypes like usability or reliability are often ambiguously phrased (e.g., "The system shall be efficient" could apply to multiple categories), leading to lower full classification accuracies. Even BERT-based models struggled to achieve above 78.4% accuracy, suggesting inherent difficulties in the task.
- **Dataset Limitations:** The dataset's size and diversity are unknown but likely limited, given the reliance on synthetic oversampling. Small datasets may not capture the full range of requirement types, reducing generalizability.
- **Model Complexity:** BiLSTM models (6 and 7) performed best but are computationally expensive, which may limit their practical deployment in resource-constrained environments. Simpler models like Random Forest (Model 1) offer a trade-off between performance and efficiency.
- **Full Classification Difficulty:** The significant drop in performance from binary to full classification (e.g., 97.27% to 78.4% for Model 7) highlights the complexity of multi-class classification. Overlapping subtypes and insufficient training examples for certain categories likely contributed to this gap.

## 7. IMPLICATIONS FOR SOFTWARE ENGINEERING

The results have significant implications for requirements engineering:

- **Automated Classification:** Models 6 and 7, with 97.27% binary classification accuracy, could be integrated into requirement management tools (e.g., IBM DOORS, Jira) to automatically flag NFRs, reducing manual effort and improving requirement elicitation.
- **NFR Subtype Identification:** Model 7's 78.4% full classification accuracy, while not perfect, is sufficient for prioritizing critical

NFRs (e.g., security, performance) in safety-critical systems, ensuring they are addressed during design and testing.

- **Practical Trade-offs:** Simpler models like Model 1 (90.4% binary accuracy) may be preferred in resource-constrained settings due to their lower computational cost, despite slightly lower performance.
- **Quality Assurance:** Accurate NFR classification can enhance traceability and validation, ensuring that non-functional aspects like usability or reliability are not overlooked in development.

## 8. FUTURE RESEARCH DIRECTIONS

To address the identified challenges and improve performance, future research could explore:

- **Larger and More Diverse Datasets:** Collecting larger datasets with balanced representation of NFR subtypes could reduce reliance on synthetic oversampling and improve generalizability.
- **Advanced NLP Models:** Fine-tuning transformer-based models (e.g., RoBERTa, T5) or using domain-specific pre-training on software engineering texts could enhance performance, particularly for full classification.
- **Hybrid Approaches:** Combining rule-based methods (e.g., keyword matching for performance requirements) with machine learning could improve accuracy for clear-cut cases while leveraging NLP for ambiguous cases.
- **Active Learning:** Implementing active learning to prioritize annotation of uncertain or underrepresented requirements could reduce labeling costs and improve model performance.
- **Explainability:** Developing interpretable models or post-hoc explanation techniques (e.g., SHAP for Random Forest) could increase trust in automated classification, especially for critical applications.
- **Real-World Evaluation:** Testing models on real-world projects could validate their practical utility and identify deployment challenges.
- **KG evolution through continuous learning, Link prediction for missing relationships, and Integration with domain ontologies to improve semantic expressiveness.**

## CONCLUSION

The experiments demonstrate that automated classification of software requirements is feasible, with Models 6 and 7 (BERT, BiLSTM, SMOTE/SMOTE-ENN) achieving exceptional binary classification performance (97.27% accuracy) and strong full classification performance (76–78.4% accuracy). BERT's contextual embeddings, combined with effective minority class handling and sequential modeling, proved critical for handling the nuanced and often ambiguous nature of requirements. However, full classification remains challenging due to overlapping NFR subtypes and dataset limitations. Model 7, with the highest full classification accuracy, offers the most promise for practical applications, such as automating requirement analysis in software engineering tools. Future work should focus on larger datasets, advanced NLP models, and real-world validation to further enhance performance and applicability. A distinguishing feature of the framework is its use of Knowledge Graphs (KGs) to model semantic relationships among requirements.

## REFERENCES

- [1]. Ahmad K. Requirements Engineering for Human-Centered Artificial Intelligence Software [Internet]. 2023 [cited 2025 Mar 25]. Available from: [https://dro.deakin.edu.au/articles/thesis/Requirements\\_Engineering\\_for\\_Human-Centered\\_Artificial\\_Intelligence\\_Software/27193158/1/files/49714161.pdf](https://dro.deakin.edu.au/articles/thesis/Requirements_Engineering_for_Human-Centered_Artificial_Intelligence_Software/27193158/1/files/49714161.pdf)
- [2]. Salmani A. An AI-Based Human-Centered Approach to Support Multidisciplinary Requirements Engineering [Internet]. 2023 [cited 2025 Mar 25]. Available from: <http://hdl.handle.net/1880/115823>
- [3]. Aminu Umar Supervisor M, Lano K. Automated Requirements Engineering Framework for Model-Driven Development [Internet]. 2024 Dec [cited 2025 Mar 25]. Available from: [https://kclpure.kcl.ac.uk/portal/files/324984252/2025\\_Umar\\_Muhammad\\_Aminu\\_1894251\\_ethesis.pdf](https://kclpure.kcl.ac.uk/portal/files/324984252/2025_Umar_Muhammad_Aminu_1894251_ethesis.pdf)
- [4]. Papageorgiou N, Pournara D, Apostolou D, Mentzas G. Managing industrial water treatment processes knowledge with knowledge graphs. Intelligent Decision Technologies [Internet]. 2025 Feb 27; Available from: <https://journals.sagepub.com/doi/10.1177/18724981251321368>
- [5]. Hua Y, Wang R, Wang Z, Wang G, Yan Y. Knowledge graph with deep reinforcement learning for intelligent generation of machining process design. Journal of Engineering Design [Internet]. 2024 Apr 9;1–35. Available from: <https://www.tandfonline.com/doi/full/10.1080/09544828.2024.2338342>
- [6]. Raja Pulicharla M. AI-Augmented Data Lineage: A Cognitive Graph-Based Framework for Autonomous Data Traceability in Large Ecosystems. Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal) Impact [Internet]. 2025 [cited 2025 Mar 25];8(1):377. Available from: <https://philpapers.org/archive/DRMADL.pdf>
- [7]. Aggour KS, Detor A, Gabaldon A, Mulwad V, Moitra A, Cuddihy P, et al. Compound Knowledge Graph-Enabled AI Assistant for Accelerated Materials Discovery. Integr Mater Manuf Innov [Internet]. 2022 Dec 8;11(4):467–78. Available from: <https://link.springer.com/10.1007/s40192-022-00286-z>
- [8]. Fenoglio E, Kazim E, Latapie H, Koshiyama A. Tacit knowledge elicitation process for industry 4.0. Discover Artificial Intelligence [Internet]. 2022 Dec 10;2(1):6. Available from: <https://link.springer.com/10.1007/s44163-022-00020-w>
- [9]. Cheng H, Husen JH, Lu Y, Racharak T, Yoshioka N, Ubayashi N, et al. Generative AI for Requirements Engineering: A Systematic Literature Review [Internet]. 2023 [cited 2025 Mar 25]. Available from: <https://arxiv.org/abs/2409.06741>
- [10]. Rouabhia D, Hadjadj I. Enhancing Class Diagram Dynamics: A Natural Language Approach with ChatGPT. 2024 Jun 16; Available from: <http://arxiv.org/abs/2406.11002>
- [11]. Wang H. Goal-oriented framework for AI-empowered curriculum design [Internet]. Nanyang Technological University; 2024. Available from: <https://hdl.handle.net/10356/181505>
- [12]. Li S, Zhou X, Liu Y, Chen J, Guo T, Yang W, et al. Agile conceptual design and validation based on multi-source product data and large language models: a review,

- framework, and outlook. *Journal of Engineering Design* [Internet]. 2025 Mar 11;1–31. Available from: <https://www.tandfonline.com/doi/full/10.1080/09544828.2025.2476879>
- [13]. Krause F, Paulheim H, Kiesling E, Kurniawan K, Leva MC, Estrada-Lugo HD, et al. Managing human-AI collaborations within Industry 5.0 scenarios via knowledge graphs: key challenges and lessons learned. *Front Artif Intell.* 2024;7.
- [14]. Cheligeer C, Huang J, Wu G, Bhuiyan N, Xu Y, Zeng Y. Machine learning in requirements elicitation: a literature review. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* [Internet]. 2022 Oct 26;36:e32. Available from: [https://www.cambridge.org/core/product/identifier/S0890060422000166/type/journal\\_article](https://www.cambridge.org/core/product/identifier/S0890060422000166/type/journal_article)
- [15]. Schlutter A, Vogelsang A. Knowledge Representation of Requirements Documents Using Natural Language Processing. 2018; Available from: <https://doi.org/10.14279/depositoncc-7776>
- [16]. Zhu X, Li H, Su T. Autonomous complex knowledge mining and graph representation through natural language processing and transfer learning. *Autom Constr* [Internet]. 2023 Nov 1;155:105074. Available from: <https://linkinghub.elsevier.com/retrieve/pii/S0926580523003345>
- [17]. Da P, Fiorela :, Supervisor C, Presutti V, Bartolini I. SUPPORTING REQUIREMENT ELICITATION AND ONTOLOGY TESTING IN KNOWLEDGE GRAPH ENGINEERING Coordinatore Dottorato [Internet]. 2023 [cited 2025 Mar 25]. Available from: [http://amsdottorato.unibo.it/11057/1/Fiorela\\_a\\_Ciroku-PhD\\_Thesis.pdf](http://amsdottorato.unibo.it/11057/1/Fiorela_a_Ciroku-PhD_Thesis.pdf)
- [18]. Khorashadizadeh H, Zahra Amara F, Ezzabady M, Ieng F, Tiwari S, Mihindikulasooriya N, et al. Research Trends for the Interplay between Large Language Models and Knowledge Graphs. 2024 [cited 2025 Mar 25]; Available from: <https://vldb.org/workshops/2024/proceedings/LLM+KG/LLM+KG-9.pdf>
- [19]. Mahon L. Machine Learning and Knowledge Graphs: Existing Gaps and Future Research Challenges. 2023 [cited 2025 Mar 25];1(1). Available from: [https://ricerca.uniba.it/retrieve/2319ac62-212c-439a-84c8-48d8df2e5c53/TGDK23\\_GraphData\\_Kg\\_DNN\\_Vision-CameraReady.pdf](https://ricerca.uniba.it/retrieve/2319ac62-212c-439a-84c8-48d8df2e5c53/TGDK23_GraphData_Kg_DNN_Vision-CameraReady.pdf)
- [20]. Zhou Q, Li T, Wang Y. Assisting in requirements goal modeling: a hybrid approach based on machine learning and logical reasoning. In: *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems* [Internet]. New York, NY, USA: ACM; 2022. p. 199–209. Available from: <https://dl.acm.org/doi/10.1145/3550355.3552415>
- [21]. Mattioli J, Tachet D, Tschirhart F, Sohler H, Cantat L, Robert B. Leveraging Knowledge Graph to Design the Machine-Learning Engineering Body-of-Knowledge. In: *2024 Conference on AI, Science, Engineering, and Technology (AIxSET)* [Internet]. IEEE; 2024. p. 258–65. Available from: <https://ieeexplore.ieee.org/document/10770993/>
- [22]. Zhao L, Alhoshan W, Ferrari A, Letsholo KJ. Classification of Natural Language Processing Techniques for Requirements Engineering. 2022 Apr 8; Available from: <http://arxiv.org/abs/2204.04282>
- [23]. Wang L, Sun C, Zhang C, Nie W, Huang K. Application of knowledge graph in software engineering field: A systematic literature review. *Inf Softw Technol* [Internet]. 2023 Dec 1;164:107327. Available from: <https://linkinghub.elsevier.com/retrieve/pii/S0950584923001829>
- [24]. Onyeka E, Varde AS, Anu V, Tandon N, Daramola O. Using Commonsense Knowledge and Text Mining for Implicit Requirements Localization. In: *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI.* IEEE Computer Society; 2020. p. 935–40.
- [25]. Jin K, Zhuo HH. Integrating AI Planning with Natural Language Processing: A Combination of Explicit and Tacit Knowledge. 2022 Feb 14; Available from: <http://arxiv.org/abs/2202.07138>
- [26]. Dias Canedo E, Cordeiro Mendes B. Software Requirements Classification Using Machine Learning Algorithms. *Entropy* [Internet]. 2020 Sep 21;22(9):1057. Available from: <https://www.mdpi.com/1099-4300/22/9/1057>
- [27]. Rahman A, Nayem A, Siddik S. Non-Functional Requirements Classification

- Using Machine Learning Algorithms. International Journal of Intelligent Systems and Applications [Internet]. 2023 Jun 8;15(3):56–69. Available from: <https://www.mecspress.org/ijisa/ijisa-v15-n3/v15n3-5.html>
- [28]. Sonawane SN, Puthran SM. Classification of functional and nonfunctional requirements based on convolutional neural network with flower pollination optimizer. Innov Syst Softw Eng. 2024 Nov 4;
- [29]. Sabir M, Banissi E, Child M. A Deep Learning-Based Framework for the Classification of Non-functional Requirements. In 2021. p. 591–601. Available from: [http://link.springer.com/10.1007/978-3-030-72651-5\\_56](http://link.springer.com/10.1007/978-3-030-72651-5_56)
- [30]. AlDhafer O, Ahmad I, Mahmood S. An end-to-end deep learning system for requirements classification using recurrent neural networks. Inf Softw Technol [Internet]. 2022 Jul;147:106877. Available from: <https://linkinghub.elsevier.com/retrieve/pii/S0950584922000428>