

**VILNIUS GEDIMINAS TECHNICAL UNIVERSITY**

**FACULTY OF FUNDAMENTAL SCIENCES**

**DEPARTMENT OF INFORMATION SYSTEMS**

**Supun Chandimal Thilakarathna Ranasinghe Mudiyanselage (20241192)**

**Application of artificial intelligence methods in the requirements engineering management process.**

**Master Graduation Thesis**

**Guided by: Assoc. Prof. Dr. Jolanta Miliauskaitė**

VILNIUS 2025

**CONTENT**

[**INTRODUCTION** 5](#_Toc201684772)

[**Investigation Object** 6](#_Toc201684773)

[**The Aim and Tasks of the Thesis** 6](#_Toc201684774)

[**Novelty of the Topic** 7](#_Toc201684775)

[**Relevance of the Topic** 7](#_Toc201684776)

[**Research Methodology** 7](#_Toc201684777)

[**Scientific Value of the Master Thesis** 7](#_Toc201684778)

[**Main Results of the Master Thesis** 8](#_Toc201684779)

[**Structure of the Master Thesis** 8](#_Toc201684780)

[**1.** **RELATED WORKS ANALYSIS** 8](#_Toc201684781)

[**1.1.** **Main Concepts** 8](#_Toc201684782)

[**1.2 Related works on the application of artificial intelligence methods in the requirements engineering management process.** 11](#_Toc201684783)

[**1.2.1.** **Requirement Elicitation** 11](#_Toc201684784)

[**1.2.1.** **Requirement extraction** 11](#_Toc201684785)

[**1.2.2.** **Requirement classification** 12](#_Toc201684786)

[**1.2.3.** **Requirement validation** 12](#_Toc201684787)

[**1.2.4.** **Requirement Traceability** 12](#_Toc201684788)

[**1.2.5.** **Requirement documentation** 13](#_Toc201684789)

[**2.** **METHODOLOGY** 18](#_Toc201684790)

[**2.1.** **Overview of the Proposed Methodology** 18](#_Toc201684791)

[**2.2.** **Describing A Process Model Using BPMN Diagrams** 18](#_Toc201684792)

[**2.2.1.** **Requirement Elicitation** 19](#_Toc201684793)

[**2.2.2.** **Requirement Extraction** 20](#_Toc201684794)

[**2.2.3.** **Requirement Classification** 21](#_Toc201684795)

[**2.2.4.** **Requirement Validation** 22](#_Toc201684796)

[**2.2.5.** **Requirement Documentation** 23](#_Toc201684797)

[**3.** **RESULTS MEASUREMENT METRICS** 24](#_Toc201684798)

[**3.1.** **Extraction Evaluation Metrics** 24](#_Toc201684799)

[**3.2.** **Classification Metrics** 25](#_Toc201684800)

[**3.3.** **Validation and Quality Assurance Metrics** 25](#_Toc201684801)

[**4.** **RESULTS ACHIEVED DURING INITIAL EXPERIMENTATION** 27](#_Toc201684802)

[**4.1.** **Requirement Extraction Results (Zero-Shot)** 27](#_Toc201684803)

[**4.2.** **Requirement Classification Results (Zero-Shot)** 28](#_Toc201684804)

[**4.3.** **Requirement Extraction (Fine-Tuned on REMBERT-AGR)** 29](#_Toc201684805)

[**4.4.** **Requirement Classification (NoBERT & PROMISE NFR)** 30](#_Toc201684806)

[**5.** **GENERAL CONCLUSION** 31](#_Toc201684807)

[**6.** **REFERENCES** 31](#_Toc201684808)

**LIST OF FIGURES**

[Figure 1. High-level activity model of the RE process (Kotanya & Sommerville, 1998) 9](#_Toc201684577)

[Figure 2. Requirement Extraction from User Stories 9](#_Toc201684578)

[Figure 3. The two steps of how BERT is developed (The Illustrated BERT, ELMo, and Co. (How NLP Cracked Transfer Learning) – Jay Alammar – Visualizing Machine Learning One Concept at a Time., n.d.) 10](#_Toc201684579)

[Figure 4. BPMN Diagram for Proposed Methodology Main Process 19](#_Toc201684580)

[Figure 5. Requirement Extraction sub process. 21](#_Toc201684581)

[Figure 6. Requirement Classification sub process. 22](#_Toc201684582)

[Figure 7. Requirement Validation sub process. 23](#_Toc201684583)

[Figure 8. AGR Extraction and Slot Completness 27](#_Toc201684584)

[Figure 9. Weighted Confidence score and manual review 27](#_Toc201684585)

[Figure 10. Sample requirement graph visualization and table visualization 28](#_Toc201684586)

[Figure 11.Predicted class distributions and confidence values 29](#_Toc201684587)

**LIST OF TABLES**

[Table 1. Summary of AI Techniques and Tools for Optimizing Requirements Engineering Phases 13](#_Toc201037514)

[Table **2.** Data extraction template for research on applying Artificial Intelligence in the Requirements Engineering Management Process. 15](#_Toc201037515)

**ABBREVIATIONS**

The abbreviations used in this document are described as follows:

AI: Artificial Intelligence

ML: Machine Learning

NLP: Natural Language Processing

RE: Requirements Engineering

SLR: Systematic Literature Review

MBSE: Model-Based Systems Engineering

UML: Unified Modeling Language

SysML: Systems Modeling Language

IoT: Internet of Things

LLM: Large Language Model

BERT: Bidirectional Encoder Representations from Transformers

SVM: Support Vector Machine

NFR: Non-Functional Requirements

FR: Functional Requirements

EPS: Engineered Process Steps

BPR: Business Process Reengineering

SE: Software Engineering

KPCA: Kernel Principal Component Analysis

AL: Adversarial Learning

GABP: Genetic Algorithm Backpropagation

SDLC: Software Development Life Cycle

REMP: Requirements Engineering Management Process

# **INTRODUCTION**

The problem of managing requirements in software engineering remains a critical challenge, as uncommitted, incomplete, or inconsistent requirements can lead to project failures, cost overruns, and delays in delivery (Kotanya & Sommerville, 1998). Requirement engineering (RE) plays an important role in the software development lifecycle (SDLC) by defining, analyzing, documenting, and validating system requirements in order to ensure compliance with business objectives and the expectations of stakeholders (Chazette et al., 2021). However, traditional RF methods are often inefficient, take time, and often involve manual processes such as interviews, surveys, and document reviews (Okonkwo & Igah, 2023). Furthermore, the requirements must evolve in order to adapt to changing business needs, but traditional methods fail to deal effectively with changes, which leads to poor requirement traceability and increased project risk. Due to the different interpretations of stakeholders, the complexity of the evolving system, and the manual dependence tracking, it is difficult to maintain accuracy, traceability, and completeness throughout the development cycle, causing the problem of inconsistency, incompatibility and ambiguity of requirements. Thus, it is crucial to explore innovative approaches such as artificial intelligence (AI) to improve the efficiency and accuracy of the requirements engineering management process (REMP).

The problem of generating requirements is even more complicated by the increasing complexity of modern software systems and the dynamic nature of the needs of stakeholders. As the scale of software projects increases, manually collecting and analyzing requirements becomes unfeasible, leading to a lack of requirements, conflicting stakeholder expectations, and undefined functional and non-functional needs (Abdelnabi et al., 2021). In addition, the participation of multiple stakeholders from a variety of perspectives increases the likelihood of discrepancies in the interpretation of requirements. Manual methods are often insufficient for handling large-scale, complex projects, and frequently require changes, as they lack scalability. This lack of adaptability led to a shrinking of requirements, which had a negative impact on the scope and budget of the project (Binder & Mezhuyev, 2024). Furthermore, inconsistencies between requirements specifications and actual system implementation often lead to design flaws and software errors, leading to increased maintenance costs. To solve these problems, it is necessary to adopt an automated solution that intelligently extracts, classifies, and validates requirements in real-time, reduces inconsistencies, and improves the alignment of projects with the objectives of stakeholders.

The problem of requirement validation and consistency checking is another significant challenge in Requirements Engineering Management. Traditional methods rely on human analysts to detect ambiguities, contradictions, and missing information, which is both labor-intensive and prone to errors (Prifti, 2022). Ensuring that requirements are well-defined and do not conflict with each other is critical for maintaining software quality and minimizing rework costs. Manual validation processes often fail to capture hidden dependencies among requirements, increasing the likelihood of defects in later development stages. Recent advancements in Natural Language Processing (NLP) and Machine Learning (ML) have introduced automated techniques that can analyze requirement documents, detect inconsistencies, and suggest refinements (Salmani, 2023). These AI-driven tools can assist in requirement verification, compliance checking, and impact analysis, enhancing the overall accuracy of the requirement specification process. However, the integration of AI-based tools into RE management remains an evolving field, requiring further research to optimize AI's role in improving requirement completeness, traceability, and adaptability in large-scale projects.

The problem of inefficient requirement management can be addressed by integrating Artificial Intelligence (AI) techniques into the Requirements Engineering Management Process (REMP). AI-powered solutions, such as Natural Language Processing (NLP), Machine Learning (ML), and transformer-based models (e.g., BERT, GPT-4), offer promising approaches for automating requirement extraction, classification, ambiguity detection, and validation(Okonkwo & Igah, 2023). AI-driven requirement traceability systems can also enhance project alignment by linking requirements across different phases of the SDLC, ensuring consistency and minimizing the risk of misinterpretation (Prifti, 2022). Additionally, AI can provide predictive analytics, enabling early detection of potential requirement conflicts and suggesting optimizations based on historical project data. By leveraging AI, organizations can improve the accuracy, efficiency, and adaptability of requirement management processes, ultimately leading to more reliable and cost-effective software development. This study aims to explore the application of AI methods in Requirements Engineering Management, analyzing their effectiveness in enhancing requirement elicitation, classification, validation, and traceability to address the challenges faced in traditional RE approaches.

## **Investigation Object**

The investigation object of the research is the application of AI methods in Requirements Engineering Management (REMP), focusing on automating extraction, classification, and validation through NLP and machine learning to enhance efficiency, scalability, and adaptability to evolving stakeholder needs.

## **The Aim and Tasks of the Thesis**

The aim of the research is to enhance the extraction and classification of software requirements using AI techniques, particularly natural language processing (NLP) and machine learning, to improve accuracy and automation in Requirements Engineering.

The main tasks for the research are as follows:

1. To analyze existing AI applications for requirement extraction and classification, with a focus on how machine learning and NLP techniques—particularly transformer-based models—are applied to improve accuracy and automation in Requirements Engineering.
2. To propose an AI-based framework that utilizes natural language processing (NLP) and machine learning for effective extraction and classification of software requirements from natural language texts.
3. To design and implement a prototype system that demonstrates the practical application of AI methods for automating requirement extraction and classification in real-world software engineering scenarios.

## **Novelty of the Topic**

The novelty of this research lies in leveraging AI-driven automation for requirement extraction, classification in Requirements Engineering Management (REMP). Unlike traditional manual approaches, this study integrates NLP, ML, and transformer models to enhance accuracy and automation in RE throughout the Software Development Lifecycle (SDLC).

## **Relevance of the Topic**

This research is relevant as it addresses the challenges of ambiguity, inconsistency, and inefficiency in Requirements Engineering by focusing on the automation of requirement extraction and classification. Traditional manual methods often fail to cope with the complexity and variability of natural language used in stakeholder requirements. By integrating natural language processing (NLP), machine learning (ML), and transformer-based models, this research aims to enhance the accuracy, scalability, and consistency of extracting and classifying software requirements, thereby improving automation and reducing human effort in the early phases of software development.

## **Research Methodology**

The research methodology follows a systematic approach to analyzing the application of Artificial Intelligence (AI) in Requirements Engineering Management (REMP). The study employs comparative analysis to evaluate existing AI-driven techniques for requirement extraction, classification, validation, and traceability. Logical induction and generalization are used to develop a conceptual framework for integrating AI into RE processes. The methodology also includes systematization and evaluation of AI models, such as Natural Language Processing (NLP) and Machine Learning (ML), to assess their effectiveness in managing requirements. Conceptual modeling is applied to define an AI-based RE management approach. The experimental method is used for practical implementation, testing AI-driven techniques on real or simulated datasets. Additionally, a case study approach is employed to validate the proposed AI framework in a real-world software development environment. The methodology ensures a structured, data-driven evaluation of AI applications in REMP, aiming for a scalable and adaptable solution.

## **Scientific Value of the Master Thesis**

New AI-based techniques improve requirement extraction and classification using transformer models like BERT, reducing ambiguity and inconsistencies. An automated AI-driven traceability approach enhances Requirements Engineering Management (REMP) by ensuring alignment across the Software Development Lifecycle (SDLC). This research contributes new knowledge by integrating NLP and ML for a scalable, accurate, and efficient requirement management framework.

## **Main Results of the Master Thesis**

Analysis of related literature and existing AI-driven approaches in requirements engineering reveals that traditional methods often struggle with ambiguity, incompleteness, and overlapping categories, making manual requirement extraction and classification inefficient and prone to error. This research demonstrates that transformer-based models, such as BERT, significantly enhance the accuracy and automation of requirement extraction and classification tasks. A critical evaluation of current AI-based methods highlights ongoing challenges in scalability, adaptability to diverse domains, and seamless integration into engineering workflows. To address these issues, a novel AI-powered framework is proposed, leveraging NLP and machine learning techniques to optimize requirement classification and extraction. Experimental results confirm that the proposed approach outperforms traditional manual processes, leading to improved accuracy, efficiency, and reduced human effort in requirements management.

## **Structure of the Master Thesis**

The introduction defines the problem, objectives, and scope of the research. Analytical Part reviews more than 20 scientific articles on AI applications in requirements engineering (RE), identifying key concepts and approaches. The Comparison Table summarizes findings from related studies, comparing their methods, datasets, and results.

# **RELATED WORKS ANALYSIS**

## **Main Concepts**

The main concept of this research revolves around leveraging Artificial Intelligence (AI) to enhance the Requirements Engineering Management Process (REMP) by automating requirement extraction, classification, validation, and traceability. Natural Language Processing (NLP) plays a crucial role in RE by enabling machines to understand, interpret, and analyze textual data, improving accuracy and reducing manual effort (Chazette et al., 2021). NLP techniques, such as tokenization, part-of-speech tagging, and named entity recognition, facilitate requirement classification and ambiguity detection, significantly enhancing automation in RE (Okonkwo & Igah, 2023). Recent advancements in deep learning models, particularly transformer-based architectures, have further improved requirement validation and traceability(Abdelnabi et al., 2021).

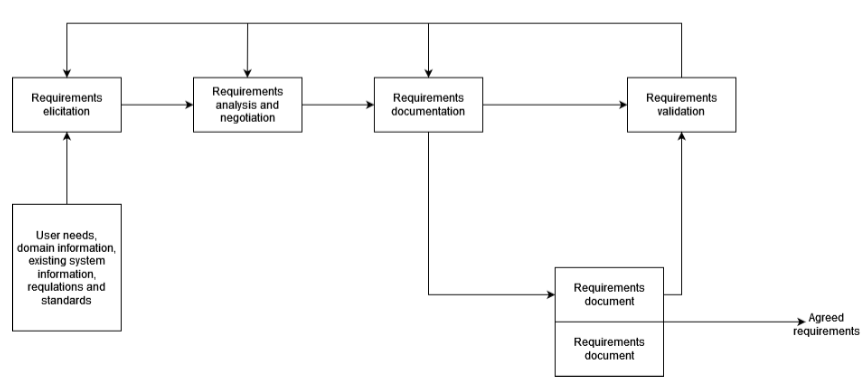
A requirement is a documented need, function, or constraint that a system must satisfy to meet stakeholder expectations, typically categorized as functional requirements (what the system should do) and non-functional requirements (how the system should perform)(Chazette et al., 2021). Requirements Engineering (RE) is a critical phase of the Software Development Lifecycle (SDLC), which is a structured process guiding software development through planning, analysis, design, implementation, testing, deployment, and maintenance (Kotanya & Sommerville, 1998). RE occurs in the planning and analysis phases of SDLC, where system requirements are elicited, analyzed, documented, and validated to ensure alignment with business goals and user needs. Figure 1 shows a High-level activity model of the RE process, adapted from (Kotanya & Sommerville, 1998). Traditionally, requirements are collected through interviews, surveys, focus groups, prototyping, and document analysis, then compiled into a Software Requirements Specification (SRS) document for reference throughout development (Okonkwo & Igah, 2023). However, the manual RE process is time-consuming, prone to human errors, and often leads to ambiguous, inconsistent, or incomplete requirements, particularly in large and complex projects (Chazette et al., 2021). These challenges contribute to scope creep, misalignment between stakeholders and developers, costly rework, and project failures (Abdelnabi et al., 2021).

Figure 1.High-level activity model of the RE process (Kotanya & Sommerville, 1998)

The requirements extraction process is a crucial step in Requirements Engineering (RE), ensuring that software projects align with stakeholder needs and system objectives. This process begins with collecting user stories, which describe system functionalities from an end-user perspective (Kotanya & Sommerville, 1998). These user stories undergo preprocessing, including tokenization, part-of-speech (POS) tagging, and lemmatization, to standardize textual data for analysis (Chazette et al., 2021). Key components actor, action, and object are then extracted to structure requirements systematically (Okonkwo & Igah, 2023). Following extraction, the requirements are classified into Functional Requirements (FR), defining system behavior, and Non-Functional Requirements (NFR), specifying quality attributes such as security and performance (Abdelnabi et al., 2021). Finally, these structured requirements are stored in a formalized format (e.g., CSV, database) to ensure traceability and facilitate further validation (Akbar et al., 2021). Figure 2 shows the Requirement Extraction from User Stories. This structured approach enhances requirement consistency, minimizes ambiguity, and supports efficient requirement management throughout the software development lifecycle.

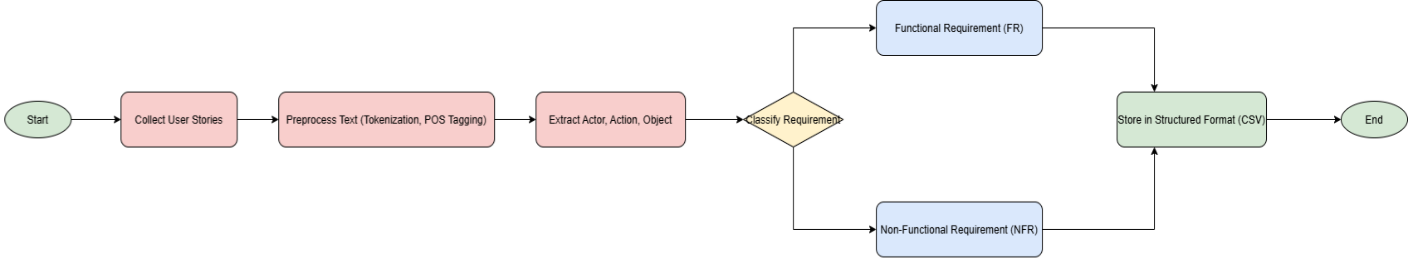


Figure 3. Requirement Extraction from User Stories

Natural Language Processing (NLP) employs several core techniques to process textual data efficiently, including tokenization, part-of-speech (POS) tagging, and named entity recognition (NER). Tokenization is the process of splitting text into smaller units, such as words or subwords, to facilitate computational processing and linguistic analysis, making it a fundamental step in NLP-based requirements extraction (Okonkwo & Igah, 2023). Part-of-Speech (POS) tagging assigns grammatical categories like nouns, verbs, and adjectives to each token, helping AI models interpret sentence structure and improve accuracy in requirement classification (Kaur & Kaur, 2024). Named Entity Recognition (NER) identifies and categorizes key entities such as persons, organizations, locations, and dates, which is crucial for extracting structured information from unstructured textual requirements (Van Remmen et al., 2023b). These techniques collectively enhance requirement extraction, classification, and validation in software development, allowing AI-driven models to reduce ambiguity, improve automation, and enhance traceability in Requirements Engineering Management.

Artificial Intelligence (AI) has emerged as a transformative solution in Requirements Engineering by automating key processes such as requirement extraction, classification, and validation. AI-driven methods leverage Natural Language Processing (NLP), Machine Learning (ML), and deep learning models to efficiently analyze large volumes of required texts (Akbar et al., 2021). AI in Project Management (PM) also improves requirement traceability by linking requirements across different Software Development Lifecycle (SDLC) phases, ensuring better project alignment and decision-making (Prifti, 2022). Despite its benefits, AI-based RE methods face challenges such as data quality issues, model interpretability, and integration complexities, requiring further optimization to maximize their effectiveness in software development environments (Abdelnabi et al., 2021).

Transformer-based models, such as Bidirectional Encoder Representations from Transformers (BERT), have revolutionized NLP by enabling a context-aware understanding of text. Unlike traditional NLP models that process words sequentially, transformers analyze text bi-directionally, capturing deeper linguistic relationships and improving accuracy. BERT is pre-trained on vast textual datasets using masked language modeling, allowing it to understand semantic nuances and dependencies. In Requirements Engineering (RE), BERT is fine-tuned for requirement classification, ambiguity detection, and traceability, significantly reducing human intervention in requirement analysis (Okonkwo & Igah, 2023). Figure 3 shows two steps followed when building BERT LLM.

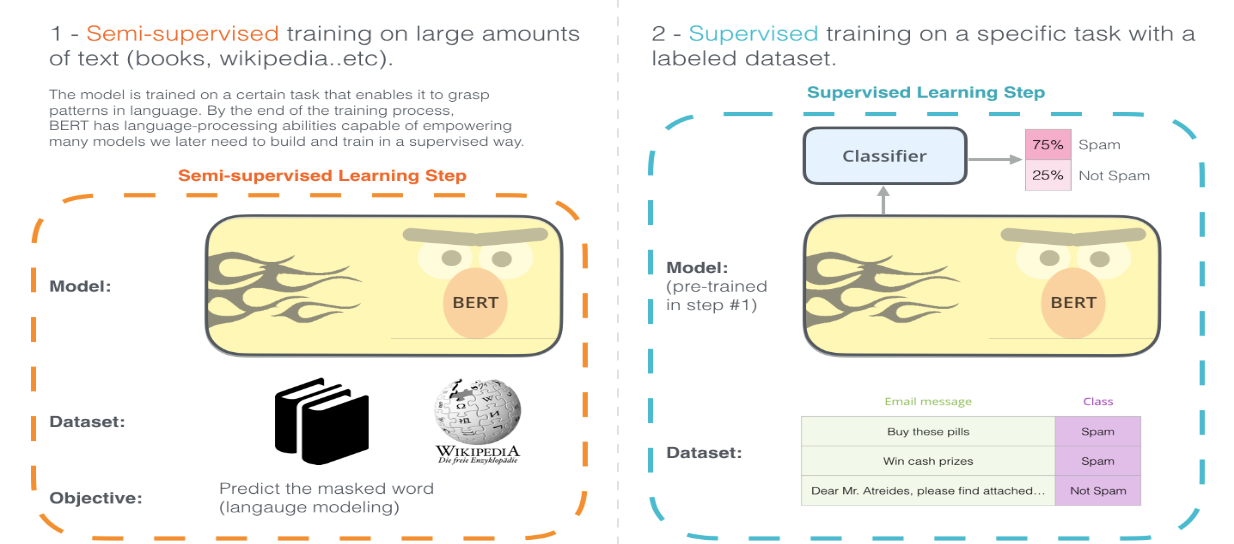


Figure 4. The two steps of how BERT is developed (The Illustrated BERT, ELMo, and Co. (How NLP Cracked Transfer Learning) – Jay Alammar – Visualizing Machine Learning One Concept at a Time., n.d.)

Validation in Requirements Engineering ensures that documented requirements accurately reflect stakeholder needs and system expectations. Traditional validation methods involve manual reviews, stakeholder discussions, and prototyping, which are often subjective and error-prone (Kotanya & Sommerville, 1998). AI-driven validation approaches utilize Machine Learning algorithms and NLP techniques to detect ambiguities, inconsistencies, and missing information, improving requirement accuracy and completeness (Akbar et al., 2021). AI-enhanced validation processes streamline software development, ensuring that validated requirements are consistent, well-defined, and traceable throughout the Software Development Lifecycle (SDLC) (Prifti, 2022).

## **1.2 Related works on the application of artificial intelligence methods in the requirements engineering management process.**

### **Requirement Elicitation**

The application of artificial intelligence (AI) in requirements elicitation has gained momentum as researchers seek to address the inefficiencies and ambiguities of traditional elicitation methods. Traditional techniques such as interview, focus groups, and document analysis are often limited by scale and subjectivity (Rehman et al., 2013). In order to overcome these challenges, recent research has introduced AI-driven solutions that use natural language processing (NLP), machine learning (ML) and deep learning (DL) to automate or enhance the induction process.(Sonbol et al., 2022) carried out a systematic mapping review showing how text representations based on NLP, including integration and syntactic parsing, are used to extract and analyze requirements from natural language documents. (Alhoshan et al., 2023) proposed a zero-shot learning approach to classify requirements (e.g. functional vs. non-functional) without marking training data, demonstrating that AI can reduce dependence on domain-specific annotations. Similarly, (Zhou et al., 2022) introduced a hybrid AI framework that combines logical reasoning and DL to support interactive goal modeling from text requirements and bridge the gap between the input of the primary stakeholders and the structured model. (Muhamad et al., 2023) used ensemble learning methods to detect ambiguity and error risk in software requirements (SRS) specifications, enhancing initial validation. These work collectively highlight the shift towards data-based, scalable, and semi-automated requirements-driven requirements-elicitation processes, while recognizing the continuing need for human-in-the-loop supervision to manage context and domain nuances.

### **Requirement extraction**

The application of artificial intelligence (AI) methods to extract requirements has become a key research area in recent years, aimed at automating and improving the accuracy of the identification of software requirements from non-structural or semi-structural sources. Traditional extraction processes, which depend heavily on manual efforts and domain expertise, face challenges in scaling and consistency, especially when processing large volumes of text data (Rehman et al., 2013). In order to address these challenges, artificial intelligence technologies such as natural language processing (NLP), machine learning (ML) and deep learning (DL) are widely used. (Sonbol et al., 2022) Presented a systematic mapping review that identified the transition from syntactic and lexical features to advanced embedded-based representations (e.g. BERT, Word2Vec) to support various required engineering tasks, including extraction. (Alhoshan et al., 2023) proposed a zero-shot learning (ZSL) approach to classifying requirements from text without labelling training data, indicating that such models can effectively extract functional and non-functional requirements. Similarly, (Muhamad et al., 2023) developed a deep learning ambiguity classification model to extract software requirements that are susceptible to errors, improving the detection of ambiguous or unclear expressions in software requirements specifications (SRS). (Zhou et al., 2022) also contributed by proposing a hybrid methodology that combines logical reasoning with DL to extract objective requirements from natural languages. These studies collectively demonstrate that AI-powered extraction methods significantly improve efficiency, scalability, and accuracy in early software development tasks.

### **Requirement classification**

Artificial intelligence (AI) methods have greatly advanced the task of classification of requirements by automating the classification of functional and non-functional requirements, a process traditionally based on manual analysis. Recent work has introduced domain-adapted language models and deep learning techniques to improve accuracy and efficiency. One of the notable contributions was NoRBERT, proposed by (Zhou et al., 2022), a domain-specific language model refined on requirements engineering body. NoRBERT surpasses general-purpose transformers such as BERT in classifying requirements according to standards such as ISO/IEC 29148, especially under low resources. In addition, zero-shot learning models were used, to address the lack of label data (Alhoshan et al., 2023) showing how such models can effectively classify requirements without task-specific training. Other studies, such as (Kaur & Kaur, 2024), have mapped the broadest use of AI techniques, highlighting the spread of SVMs, CNNs, and teaching-transfer training in the literature. Furthermore, collaborative learning approaches such as those used by (Muhamad et al., 2023) Support a robust classification by detecting ambiguous or incomplete requirements. Together, these works reflect a clear trend towards specialized, scalable and high-performance AI solutions in the field of automated requirement classification.

### **Requirement validation**

Artificial intelligence (AI) is increasingly used to automate and improve requirements validation, to overcome the limitations of traditional manual techniques, such as inspection and inspection. Recent research has focused on the use of natural language processing (NLP) and machine learning to detect ambiguities, inconsistencies, and incomplete requirements. For example (Muhamad et al., 2023) proposed a deep learning-based model combined with joint learning to identify software requirements that are susceptible to errors and ambiguities in SRS documents. (Necula et al., 2024) emphasized the role of NLP and large-scale language models in assessing linguistic quality and identifying false or false statements. (Zhou et al., 2022) introduced a hybrid approach that integrates logical reasoning and machine learning to validate goal-oriented requirements extracted from natural language texts, bridging the gap between informal input and formal validation processes. These approaches demonstrate that AI-driven validation tools can improve accuracy, scaleability, and consistency in early software development stages.

### **Requirement Traceability**

The application of artificial intelligence (AI) for the tracking of requirements has gained momentum as researchers seek to improve the accuracy and scale of linking requirements to other software artifacts, such as code, design documents, and test cases. Traditional traceability methods often depend on manual effort, which is time-consuming and prone to human error, especially in large systems. Artificial intelligence technologies, especially natural language processing (NLP) and machine learning (ML), are introduced to automate the generation and maintenance of trace links. (Sonbol et al., 2022) emphasized how embedded representations (e.g. Word2Vec, BERT) allow semantic comparison between requirement texts and target artifacts, significantly improving the prediction of trace links. Deep learning models can capture contextual similarities beyond lexical overlaps, making them more effective in generating accurate trace links in noisy or unstructured datasets. In addition, recent studies have highlighted the use of supervised and non-supervised learning to address issues such as traceability recovery and link verification, allowing systems to dynamically update links as artifacts evolve. The tracking enabled by AI not only supports compliance and impact analysis, but also reduces the cognitive burden on software engineers to maintain the tracking link throughout the life cycle.

### **Requirement documentation**

Artificial intelligence (AI) methods have increasingly been used to support and automate the documentation of requirements to improve coherence, reduce uncertainty and accelerate the production of high-quality software specification (SRS). Traditionally, the documentation of requirements has been a manual, labour-intensive task, often subject to human error and inconsistency, especially in large-scale or distributed projects. To address these issues, researchers are using natural language processing (NLP), machine learning (ML), and large language models (LLMs) to assist in the generation, structure and refinement of requirement documents. For example, (Necula et al., 2024) discussed the use of NLP techniques to analyze and improve the clarity of natural language requirements and to facilitate the generation of structured documents from the input of informal stakeholders. Similarly, (Muhamad et al., 2023) proposed a SRS detection model that avoids faults and is equipped with ensemble learning, which not only identifies unclear and unclear requirements, but also contributes to improving the quality of documented specifications. Tools powered by AI – such as IBM’s Requirements Quality Assistant – also appeared, providing real-time feedback on the quality and structure of requirement declarations in documentation. These developments reflect a shift towards semi-automated document workflows, where AI supports engineers in producing more clear, consistent and maintainable requirements artifacts.

Table 1 summarizes AI techniques applied across five phases of the requirements engineering process. It highlights key models (e.g., BERT, NoRBERT), performance metrics (accuracy, F1-score), limitations, and practical tools. The goal is to guide BPMN modeling of an AI-optimized requirements workflow.

Table 1. Summary of AI Techniques and Tools for Optimizing Requirements Engineering Phases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RE Phase** | **AI Techniques & Models** | **Performance** | **Limitations** | **Tools/Frameworks** |
| **1. Elicitation** | Chatbots (GPT/LLM), NLP for feedback mining, AI-generated surveys | Up to 50% time saved, ~0.9 precision/recall in feedback classifiers | Needs domain context, may miss tacit info, stakeholder trust issues | ChatGPT, Dialogflow, IBM Watson, Jama, Azure DevOps |
| **2. Extraction** | Rule-based parsing, BERT/BiLSTM tagging, knowledge graph construction | 85–95% accuracy on structured stories, graph models enable querying | Struggles with story variation requires labeled data | Visual Narrator, spaCy, NLTK, Jira plugins |
| **3. Classification** | SVM, CNN, BERT-based (e.g., NoRBERT), transfer learning | 90–95% accuracy, NoRBERT: 94% F₁ (FR/NFR), ~87% F₁ (NFR types) | Domain shift, class imbalance, low explainability | PROMISE dataset, HuggingFace, Jama Advisor, OpenReq |
| **4. Validation** | Rule-based NLP, BERT for ambiguity, logic reasoning, traceability analysis | High recall (0.85–0.98), ~81% F₁ for ambiguity detection | False positives, sparse training data, complex quality formalization | QuARS, ARM, Jama Advisor, Visure, ReqSuite, BERT models |
| **5. Documentation** | GPT-3/4, BART/T5 summarization, AI trace linking, style-guided NLG | ~30% time saved, 80%+ quality scores, ~0.75 trace link precision | May hallucinate, lose nuance, privacy/security concerns | Copilot (Jira/MS), ChatGPT, Jama Advisor, IBM DOORS Next |

Table 2 provides the data extraction template for research on applying Artificial Intelligence (AI) in the Requirements Engineering Management Process (REMP), including key columns that help organize and analyze various studies. It begins with the reference column, which includes the citation of the research paper being analyzed, followed by the main research problem, outlining key issues such as requirement ambiguity, inconsistency, or inefficiency in manual processes. The used approach describes the AI techniques employed, such as Natural Language Processing (NLP), Machine Learning (ML), or transformer-based models (e.g., BERT, GPT-4), to enhance requirement elicitation, classification, validation, and traceability. The application domain specifies the field where the approach was applied, such as software development, cloud computing, or business process management, while the dataset used details the source, size, and nature of data, such as requirement specifications, project documentation, or stakeholder feedback. The attributes used for prediction list key variables, including requirement type (Functional or Non-Functional), complexity level, traceability, and ambiguity detection. The evaluation of the approach explains how its effectiveness was measured, using performance metrics such as classification accuracy, precision-recall, consistency detection, and impact on project outcomes. The comparison with other works highlights how the proposed AI-based approach performs relative to traditional manual methods or alternative automated techniques, and the result summarizes its contributions, such as improved accuracy, reduced processing time, enhanced traceability, and minimized requirement conflict

**Table 2.** Data extraction template for research on applying Artificial Intelligence in the Requirements Engineering Management Process.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Reference** | **Research Question** | **Approach** | **Domain** | **Dataset** | **Attributes** | **Evaluation** | **Comparison** | **Result** |
| (Buntak et al., 2021) | **How can AI enhance performance while addressing complexity, ethics, and workforce adaptation?** | **Qualitative study** | **AI in business management** | **Literature, case studies, reports** | **Performance, satisfaction, quality, risk, decision-making** | **Assessed efficiency, decision quality** | **Broader than technical studies** | **Improved efficiency and decision-making** |
| (Burggräf et al., 2022) | **How can NLP, ML, and MBSE enhance requirement interdependency analysis?** | **Design Science Research** | **RE in product development** | **48 sources from Web of Science & Semantic Scholar** | **UML, SysML, neural networks** | **Synthesized AI RE methods** | **Highlights need for structured method** | **Built method map for requirement chain** |
| (Bhavsar et al., 2019) | **How can AI-driven automation enhance SEM and SPM in BPR?** | **Literature + case studies** | **Software Engineering Management** | **Software project org. data** | **BPR, cost, lifecycle, product quality** | **Evaluated AI's impact on SEM/SPM** | **Compared to traditional PM** | **AI improved process automation** |
| (Dacre & Kockum, 2022) | **How can AI support complex project management decisions?** | **Mixed-method study** | **Project management** | **Interviews + survey (280+ PMs)** | **Decision-making, planning, success** | **Expert assessment of AI support** | **Practical challenges vs theory** | **AI aids in large-data and complex PM** |
| (Juura, 2024) | **How has AI been used in RE and what are the key techniques?** | **Systematic Literature Review** | **Requirements Engineering** | **14 from 851 AI-RE studies** | **NLP, transformers, RE phases** | **Effectiveness and challenges** | **Aligns with NLP/ML prior work** | **AI aids RE, but lifecycle gaps remain** |
| (Binder & Mezhuyev, 2024) | **Can ChatGPT generate structured IoT specs from customer input?** | **ISO 12207 + ChatGPT framework** | **IoT systems** | **Unstructured customer needs** | **Stakeholders, scenarios, components** | **Expert-reviewed outputs** | **Compared with survey results** | **ChatGPT works, but needs scope control** |
| (Chazette et al., 2021) | **How to integrate explainability (NFR) into RE and improve quality?** | **SLR + workshops** | **RE & Software Quality** | **229 papers** | **Explainability, accountability, compliance** | **Validated via literature & experts** | **Broader than prior fragmented work** | **Created model & catalogue** |
| (Corral et al., 2022) | **Which semantic reasoning systems help in RE?** | **Systematic Literature Review** | **Requirements Engineering** | **48 papers (29 high quality)** | **Ontologies, logic models** | **Focused on conflict resolution** | **Compared classical & semantic RE** | **Enhanced classification & consistency** |
| (Kostova et al., 2020) | **How is AI transforming RE and stakeholder interactions?** | **Literature & workshop synthesis** | **AI in RE** | **Academic literature** | **Elicitation, prioritization, classification** | **Assessed AI impact on RE practice** | **Stressed need for human-AI balance** | **RE becoming AI-supported** |
| (Umar & Lano, 2024) | **What is the impact of automated RE tools?** | **Systematic Literature Review** | **Automated RE** | **85 studies (1996–2022)** | **UML outputs, RE phases** | **Reviewed effectiveness of tools** | **Compared tool types and phases** | **Most tools are semi-automated** |
| (Dehn et al., 2023) | **How to integrate AI into RE via a structured framework?** | **EPS framework design** | **Automotive RE** | **Industry survey & project data** | **Ops, descriptions, AI roles** | **Real-world testing of EPS** | **Compared to generic RE processes** | **EPS improves integration clarity** |
| (Md Siddique, 2022) | **How to apply ML/NLP in RE to improve quality?** | **ML + NLP + reasoning** | **Software RE** | **Feedback, docs, comms** | **Priorities, changes, feedback** | **Precision, recall, F1, user studies** | **Outperforms manual RE** | **AI boosts speed and quality** |
| (Md Siddique, 2023) | **How does RE automation support SDLC improvement?** | **Agile + AI/NLP integration** | **RE in SDLC** | **Docs, interviews, feedback** | **Objectives, dependencies** | **Iterative RE cycles tracked** | **Better than traditional RE** | **Improves collaboration and speed** |
| (Sofian et al., 2022) | **What AI techniques are used in software engineering?** | **Systematic Mapping Study** | **Software Engineering** | **Studies (2015–2021)** | **ML types, metrics, SE phases** | **Mapped AI to SE activities** | **Broader than single-tech studies** | **ML dominates across SE, especially RE** |
| (Okonkwo & Igah, 2023) | **How can LLMs improve FR/NFR classification in RE?** | **Fine-tuned BERT variants** | **RE classification** | **PROMISE dataset (625 entries)** | **FR/NFR labels, subcategories** | **Accuracy, learning rate, overfitting** | **BERT vs SVM, Naive Bayes** | **BERT best, but tuning needed** |
| (Marques et al., 2024) | **What are ChatGPT’s benefits and risks in RE?** | **Literature review** | **Software Requirements** | **IEEE, ACM, Google Scholar** | **Prompting, ethics, hallucination** | **Comparative analysis** | **Compared with manual elicitation** | **GPT helps, but needs human oversight** |
| (Kaur & Kaur, 2024) | **How effective is AI in requirement classification?** | **Systematic Literature Review** | **RE (Analysis/Classification)** | **PROMISE, DOORS, EHR** | **Embeddings (BERT, RoBERTa)** | **Accuracy, precision, recall** | **Compared transformer vs ML** | **Transformers effective but costly** |
| (Van Remmen et al., 2023a) | **How reusable are requirement types for modeling tasks?** | **SLR (PRISMA guidelines)** | **Mechanical Engineering RE** | **26 studies (Scopus)** | **FRs/NFRs, object mappings** | **Analyzed reuse of RE classification** | **Software-centric limits in mechanical domain** | **Need domain-specific, multilingual models** |

**Summary of research papers based on Applications of application of artificial intelligence methods in the requirements engineering management process.**

The primary focus of this research is on how AI can enhance the requirements engineering process by improving automation, accuracy, and efficiency. Traditional methods are time-consuming and prone to errors, making AI-driven solutions crucial for streamlining tasks like elicitation, validation, and prioritization. AI models are proposed to reduce manual effort and handle complex, evolving requirements more effectively.

The approach involves the use of systematic literature reviews, case studies, surveys, and machine learning models such as ResNet-18 and BERT for text classification and requirements extraction. Additionally, Design Science Research (DSR) is used to develop AI-based methodologies, and the literature review helps in analyzing industry trends and previous research to validate the effectiveness of AI in RE.

The research is conducted in multiple fields such as healthcare, construction, finance, and software development to improve requirements management. In healthcare, AI aids in managing electronic health record (EHR) requirements, while in construction, it helps with complex project requirements and risk management. In software development, AI tools are used to automate requirement traceability and improve stakeholder alignment throughout the project lifecycle.

The research utilizes a mix of publicly available and domain-specific datasets to train AI models for requirements engineering tasks. Datasets like MS COCO for image-based analysis, PROMISE for software engineering, and EHR for healthcare applications are used, along with custom datasets tailored for tasks like software requirement classification and label quality assurance to ensure high-quality training data.

The attributes used for this research are such as textual features like word embeddings, semantic similarity, and requirement dependencies to improve prediction accuracy and automate classification. Metadata, including requirement complexity, historical changes, and stakeholder priorities, further refines model predictions, enabling better decision-making and alignment across project teams.

The effectiveness of AI-based approaches is evaluated through performance metrics like Precision, Recall, Accuracy, F1-Score, and AUC, which measure classification effectiveness and prediction confidence. Inter-annotator agreement is used to assess human-AI consistency, while computational efficiency is analysed to determine the scalability and practicality of AI solutions in real-world applications.

The research includes a comparative analysis with AI methods are compared with traditional approaches like manual reviews and rule-based systems to highlight their advantages in terms of accuracy, efficiency, and scalability. Machine learning models like ResNet-18 and BERT demonstrate superior performance in classification tasks, handling large volumes of requirements more effectively than conventional methods, and adapting to evolving project needs.

The results of the research indicate AI significantly improves decision-making, efficiency, and overall quality in the requirements engineering process. Advanced models like BERT, combined with natural language processing (NLP), enhance the accuracy of requirement classification, reduce human errors, and streamline the entire RE process, aligning stakeholders and optimizing workflows**.**

# **METHODOLOGY**

## **Overview of the Proposed Methodology**

This thesis proposes a modular and AI-enhanced methodology that targets the automation and accuracy improvement of two key phases in Requirements Engineering (RE): requirement extraction and requirement classification. The method is designed around the integration of modern natural language processing (NLP) models and machine learning (ML) classifiers, embedded within a broader BPMN-modeled RE workflow for traceability and process alignment. While the entire RE cycle is represented for completeness, this methodology focuses implementation and evaluation efforts specifically on the transformation of unstructured stakeholder inputs into validated and categorized requirement artifacts.

The proposed methodology begins with a requirement elicitation phase, where stakeholder inputs are collected through various channels such as interviews, chatbot-based dialogues, or textual feedback from tickets, reviews, and forms. These elicited inputs—often unstructured and variable in clarity—form the raw material for subsequent automated processing.

The methodology then focuses on two core phases: requirement extraction and requirement classification, which are implemented and evaluated in detail. In the extraction phase, elicited texts are processed using a BERT-based named entity recognition (NER) model to extract structured requirement components: Actor, Goal, and Rationale. These components are normalized and organized into a semantic requirement graph to enable traceability and quality checks. In the classification phase, each structured requirement is categorized as either functional (FR) or non-functional (NFR), with subtyping for NFRs using a fine-tuned transformer model (NoRBERT). A confidence threshold mechanism ensures that low-confidence classifications are redirected to human review, balancing automation with reliability.

This methodology emphasizes automation, accuracy, and scalability, integrating formal metrics, model confidence gates, and optional human-in-the-loop validation. Although the broader RE lifecycle is modeled using BPMN for completeness, the technical innovation is centered on the transformation of natural language inputs into structured and validated software requirements through modern AI techniques.

## **Describing A Process Model Using BPMN Diagrams**

The BPMN representation of the updated methodology (Figure 4) centers around a unified main process that governs five interconnected phases of the requirements engineering workflow: Requirement Elicitation, Requirement Extraction, Requirement Classification, Requirement Validation, and Requirement Documentation. With the research aim focusing on enhancing extraction and classification using AI—particularly NLP and machine learning—greater emphasis is placed on the internal mechanisms of these two phases. Each phase is modeled as a collapsed subprocess, supporting modular analysis and scalability. The workflow initiates with stakeholder input and feedback mining, proceeding into automated extraction where dependency parsing and transformer-based models identify roles, actions, and goals within natural language inputs. Classification then applies fine-tuned language models (e.g., NoRBERT, BERT) to accurately assign functional or non-functional labels.

Transformer models, particularly BERT and its RE-specific variant NoRBERT, are used in this methodology for both extraction (NER) and classification (FR/NFR). Transformers work by encoding input text into contextual embeddings using multi-head self-attention mechanisms, which allow the model to weigh the importance of each token in context (Vaswani et al., 2017). In extraction, BERT is fine-tuned for sequence labeling, identifying slot-level tags (e.g., Actor, Goal) from natural language. For classification, NoRBERT applies a classification head over the pooled representation to predict requirement categories. These models significantly outperform classical methods by capturing long-range dependencies and nuanced semantics, achieving F1-scores exceeding 90% in RE tasks (Okonkwo & Igah, 2023). Their integration directly supports the research aim by increasing automation and interpretability in software requirements processing.

Within the BPMN model, strategic gateways are integrated at key decision points in these phases to assess the confidence level of AI-generated outputs. When the confidence of extraction (e.g., slot-filling completeness) or classification (e.g., FR/NFR label certainty) falls below a defined threshold (e.g., 90%), the process triggers a human-in-the-loop review to ensure precision and semantic accuracy. This conditional routing maintains a balance between automation and expert validation. The final output of the modeled process is a validated and categorized set of requirements, enriched with metadata such as classification scores, traceability links, and ambiguity flags. This BPMN-driven structure enhances automation and accuracy in RE by embedding intelligent decision control into the core information transformation stages of extraction and classification.

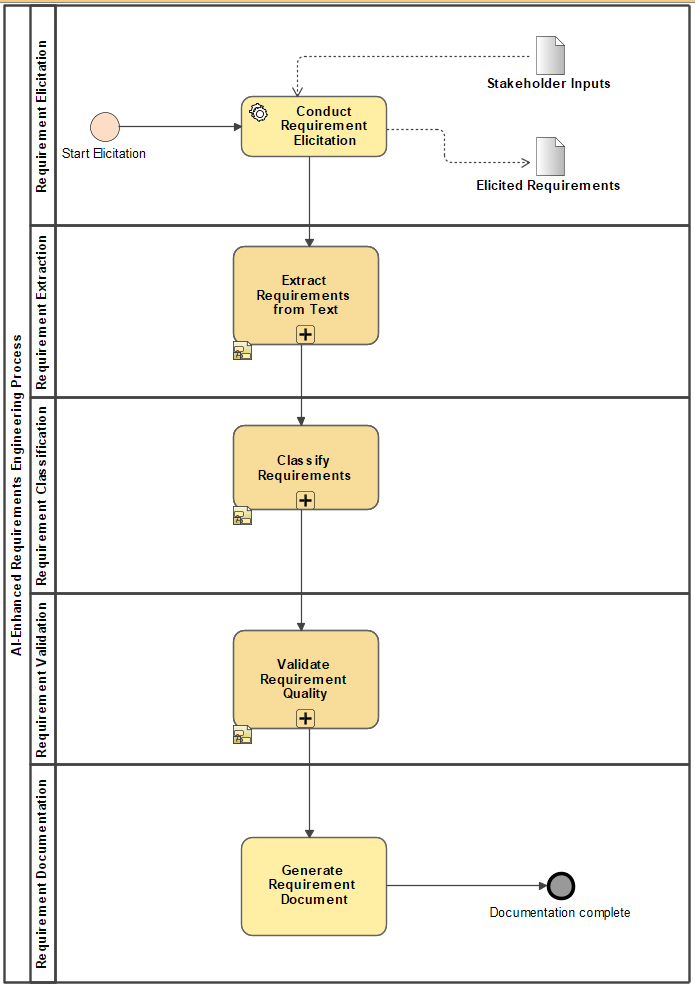


Figure 5. BPMN Diagram for Proposed Methodology Main Process

### **Requirement Elicitation**

The requirement elicitation phase in this methodology serves as the entry point for gathering stakeholder needs in natural language form. Rather than relying solely on traditional elicitation techniques such as interviews or focus groups, this process integrates multiple input channels including stakeholder feedback emails, chatbot interactions with large language models (LLMs), issue reports, and app reviews. These inputs are aggregated as unstructured textual data and form the raw material for downstream AI-based processing. The aim is to collect comprehensive and diverse perspectives from end-users and stakeholders without requiring immediate formalization. This helps to reduce the early-stage loss of information and tacit knowledge, a limitation frequently noted in conventional requirements engineering approaches(Fazelnia et al., 2024).

Empirical studies have shown that user-generated content—such as support tickets and customer reviews—can provide valuable insights into both functional and non-functional requirements when properly mined and interpreted (Sonbol et al., 2022). This methodology builds on such findings by incorporating automated preprocessing pipelines and LLM-assisted clarification to reduce ambiguity and fill information gaps before requirements are passed to the extraction phase. Clustering techniques, such as TF-IDF combined with k-means, are employed to group similar stakeholder statements and eliminate redundancy. These pre-processed and clarified statements are then treated as **elicited requirements**, structured enough for named entity recognition and requirement classification in subsequent phases.

### **Requirement Extraction**

The requirement extraction subprocess transforms elicited stakeholder input—such as feedback, chat logs, and user stories—into structured requirement components. This structured output feeds into subsequent classification and validation processes. The subprocess is modularized into key tasks designed to improve extraction accuracy, ensure semantic consistency, and support scalable automation.

The process begins with Text Preprocessing and Normalization, which standardizes the linguistic structure of inputs. This step includes lowercasing, removal of non-informative tokens, and phrase normalization using regular expressions and spaCy pipelines. Normalization ensures that expressions such as “want to login” and “need access” are mapped to consistent action phrases. This increases the downstream model’s ability to recognize the Actor–Goal–Rationale slots in a stable form and mitigates issues introduced by informal, ambiguous, or domain-specific phrasing.

The normalized text is then processed using a Dual NER Strategy. Two NER models—a fine-tuned BERT variant (e.g., NoRBERT or BERT-base-cased) and a secondary extractor (spaCy or zero-shot NER)—operate in parallel. This redundancy increases reliability, particularly in low-resource settings or when handling noisy inputs. If the two models disagree on slot boundaries, the Structure Correction step is triggered. This component uses syntactic parsing (e.g., dependency trees) to re-evaluate the phrase structure and harmonize misaligned extractions. This correction layer reduces propagation of errors to classification or documentation phases.

In this methodology, Named Entity Recognition (NER) is employed in the requirement extraction phase to identify and segment key components—namely the “Actor,” “Goal,” and “Rationale”—from unstructured textual inputs such as user stories and stakeholder feedback. A transformer-based model like BERT is fine-tuned to perform this task by labeling each token in the input text with semantic tags (e.g., B-ACTOR, I-GOAL), allowing structured triplets to be derived from natural language. This process replaces rigid template-based extraction methods, enabling more flexible handling of informal or varied sentence structures. The NER output is subsequently used to populate a semantic requirement graph for traceability and downstream analysis. The technique is implemented using Hugging Face Transformers and optionally supported by spaCy for rule-based fallback, achieving high accuracy with reported F1-scores exceeding 90% on well-annotated datasets (Fazelnia et al., 2024).

The extraction quality is measured using standard evaluation metrics such as precision, recall, and F1-score at the slot level (Actor, Goal, Rationale). Completeness of extraction is calculated based on whether all required slots are filled. Model confidence is also computed for each slot and used in downstream decision-making.

Next, the structured output is transformed into a Semantic Requirement Graph using tools like Neo4j. In this graph, nodes represent requirement entities (actor, goal, rationale), while edges represent their semantic relations. This format enables advanced reasoning, traceability, and reuse in downstream applications such as validation or stakeholder analysis.

For extractions with low confidence or incomplete slots, the instance is directed to the Active Learning Buffer. This stores uncertain examples and prompts human review. Once corrected, these instances are reused to fine-tune the model via incremental training, improving performance over time. This loop supports scalability and continuous learning, directly aligning with the thesis aim to improve automation while preserving quality.

.

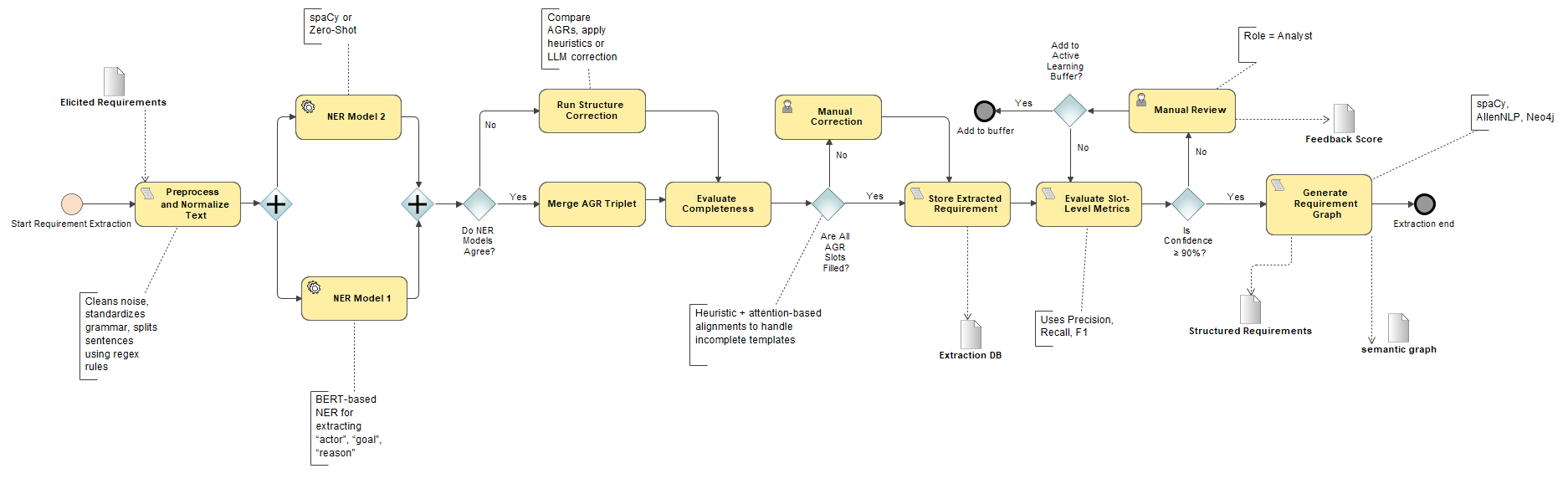


Figure 6. Requirement Extraction sub process.

### **Requirement Classification**

Following the extraction phase, the classification subprocess transforms structured requirements into enriched, multi-dimensional representations. This BPMN subprocess introduces a novel architecture that classifies each requirement along multiple dimensions—intent, quality attribute, and dependency level—using a fine-tuned BERT-based model. Unlike traditional binary classifiers that rely on single-label decisions (e.g., functional vs. non-functional), this approach enables multi-label classification through a sigmoid activation function, allowing overlapping concerns (e.g., a requirement being both performance- and security-related).

In the context of multi-label requirement classification, each requirement can belong to multiple classes simultaneously—for instance, a single requirement might pertain to both performance and security. To handle this, the model employs sigmoid activation at the output layer, enabling it to produce independent probability estimates for each class. To supervise this learning process, thesis adopt the Binary Cross-Entropy (BCE) loss function, which treats each label independently and penalizes incorrect predictions at the class level.

This formulation ensures that the model minimizes the loss for each label independently, encouraging it to assign high confidence to relevant labels while suppressing irrelevant ones. Unlike categorical cross-entropy (used for mutually exclusive classes), BCE is ideal for overlapping requirement categories common in real-world scenarios (Kaur & Kaur, 2024; Okonkwo & Igah, 2023). By combining BCE with confidence filtering and attribution mapping (e.g., SHAP, LIME), the proposed methodology not only achieves high classification accuracy but also ensures interpretability, scalability, and trustworthiness in AI-assisted requirements engineering pipelines.

This enables precise probabilistic estimates for each requirement class. The input to this classifier is a semantically structured requirement obtained from the previous extraction phase. For model training and benchmarking, we use datasets such as the PROMISE NFR Dataset, the NoBERT Requirements Corpus, and custom-annotated samples from user stories. These datasets provide balanced representations of various quality attributes (e.g., performance, maintainability, security).

To enhance trust and interpretability, the classification output is supplemented with attribution maps generated using SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations). These methods explain which parts of the input text most influenced each label, thereby promoting stakeholder transparency and enabling traceability in regulated settings.

Following classification, a confidence evaluation gateway computes the weighted average confidence score across predicted labels. This is defined as:

where is the predicted probability for the ith label, and is a domain-specific importance weight. If , the requirement is routed for manual label review. These low-confidence cases are also stored in an active learning buffer for future retraining, enhancing the model’s adaptability to unseen or ambiguous inputs.

Next, the classified requirement is integrated into a semantic requirement graph and checked for inter-label conflicts (e.g., contradictory dependencies or mutually exclusive intents). These relationships are explored using graph-based reasoning with tools like Neo4j. To detect label outliers, unsupervised clustering methods such as DBSCAN or HDBSCAN are applied. Requirements identified as outliers undergo secondary review before consolidation.

The process concludes with a ‘Consolidate Classification Output’ task, which generates the final validated, multi-label requirement object. This object includes prediction scores, attribution evidence, confidence metrics, and label interdependencies. This classification subprocess meaningfully contributes to the research aim by improving automation, transparency, and scalability in requirements engineering classification through advanced, interpretable AI techniques.

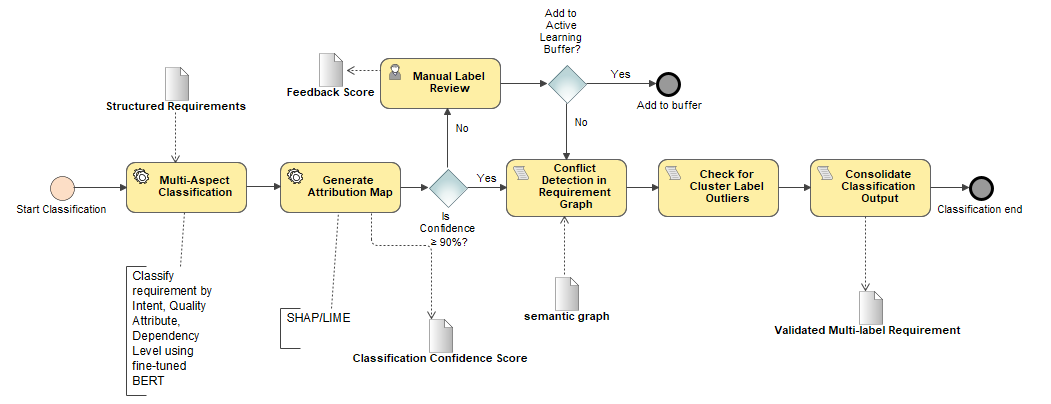


Figure 7. Requirement Classification sub process.

### **Requirement Validation**

The validation subprocess is designed as a hybrid assurance mechanism to verify the structural integrity, semantic correctness, and classification confidence of automatically processed requirements. It begins with the Self-Validation Initiation task, which retrieves structured, multi-label requirements produced during classification, along with their associated probability scores (softmax outputs). The process continues with a Cross-Perspective Validation Check, where a semantic requirement graph—generated using tools such as Neo4j or AllenNLP—is employed to analyze the logical alignment between the “actor”, “goal”, and “rationale” components. This alignment is quantified using an attention-based consistency measure derived from transformer attention weights, allowing detection of semantically disjoint assertions that might escape classification confidence alone.

To make a final decision, the system computes a Combined Confidence Score using a weighted sum of three components: classification confidence (Cc), rationale consistency (Rc), and traceability graph coherence (Gc). The final confidence is calculated as:

where w1​+w2​+w3​=1

This modular fusion of metrics represents a novel addition beyond the binary softmax thresholding typically used in RE validation. If ≥0.9, the requirement passes validation; otherwise, it is sent for Manual Review. During this user task, analysts examine logical inconsistencies, incomplete rationales, or overlapping intent types not detected by the model. Their revisions are recorded and passed to Final Validation, which logs error types and generates a Feedback Score based on the deviation between model output and human correction using a penalty function like:

2

All validated data—automatically or manually processed—is then added to an Active Learning Feedback Buffer, enabling future model fine-tuning via uncertainty sampling or disagreement-based querying.

This subprocess is distinctive in its integration of semantic traceability validation into classification assurance, going beyond the standard reliance on probabilistic thresholds. While prior literature has addressed slot-filling completeness or confidence scores in isolation, the proposed approach fuses structural reasoning (via graphs), local explanation (via attribution maps or attention weights), and human input into a unified validation framework. By doing so, it advances the research aim of enhancing both accuracy and automation in Requirements Engineering, while preserving trustworthiness through intelligent fallback mechanisms.

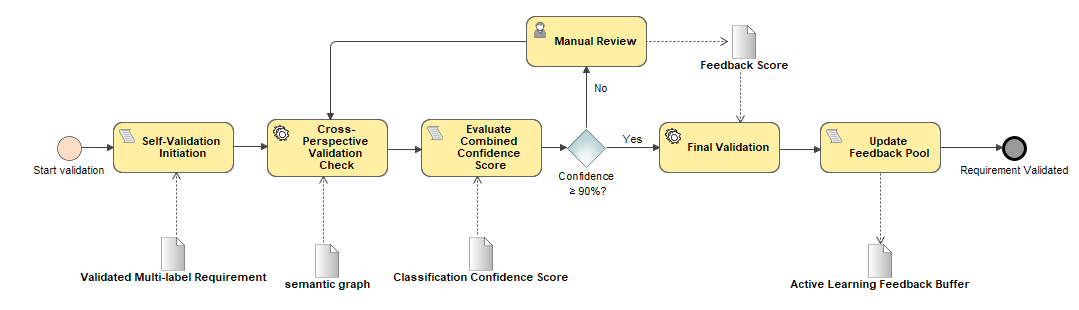


Figure 8. Requirement Validation sub process.

### **Requirement Documentation**

The Requirement Documentation task represents the final step in the AI-driven requirements engineering pipeline, transforming validated and classified requirements into a structured, human-readable format suitable for downstream development activities. Upon successful validation, each requirement—now enriched with actor-goal-rationale semantics and multi-aspect classification tags—is mapped to a documentation template such as IEEE 830, EARS, or a project-specific schema. This task uses templated generation strategies, where placeholders are populated using structured data from the requirement graph. For example, a functional requirement might be rendered as: “The [actor] shall [goal] because [rationale],” while non-functional requirements are grouped by type and linked with relevant metadata (e.g., source, timestamp, validation score). The use of this task-based documentation ensures traceability, standardization, and readability across stakeholders.

In contrast to traditional RE pipelines that rely heavily on manual editing or unstructured notes, this method automatically produces consistent requirement statements with minimal human effort. The generated documentation is stored alongside supporting artifacts such as classification confidence scores, validation logs, and semantic links. These outputs enable traceability across the lifecycle and serve as evidence for compliance audits or safety-critical reviews. By automating documentation while preserving structural integrity, this task contributes directly to the aim of improving automation and accuracy in Requirements Engineering through the integration of AI-based methods.

# **RESULTS MEASUREMENT METRICS**

This section defines the evaluation strategy for measuring the performance and reliability of the proposed methodology, which aims to improve the extraction and classification of software requirements using NLP and machine learning. The selected metrics focus on accuracy, completeness, and confidence-based automation. All metrics described here directly align with the methods and tools detailed in the earlier phases of the methodology.

## **Extraction Evaluation Metrics**

The requirement extraction phase outputs structured triplets (Actor, Goal, Rationale). Each extracted element is evaluated independently using the following standard metrics:

* Precision:

Measures how many of the extracted slot values are correct.

* Recall

Measures how many of the ground truth slot values were successfully extracted.

* F1-Score

Where TP – True Positive, FP – False Positive, FN – False Negative. A balanced metric combining precision and recall per slot.

* Completeness Rate

Evaluates how many requirements had all three slots extracted (Actor, Goal, Rationale).

* Weighted Confidence Score

Where , , and are model-provided confidence scores, and weights w₁–w₃ can be tuned based on domain importance (default: all set to 1.0). If this score is ≥ 0.90, the requirement is deemed high-confidence.

## **Classification Metrics**

Each extracted requirement is then classified as Functional (FR) or Non-Functional (NFR), and in the case of NFR, assigned one or more subtypes.

* Binary Cross-Entropy (for multi-label classification)

Where:

* + - * k is the total number of labels,
      * ∈ {0,1} is the ground truth for the label,
      * ∈ [0,1] is the predicted probability for the label.
* Precision / Recall / F1-Score

Applied on a micro and macro level to evaluate classifier performance across all labels.

* Confidence-Based Manual Review Rate
* ​ is the number of classified requirements where the predicted confidence score is below the defined threshold (e.g., 0.90),
* is the total number of classified requirements processed.

Used to measure automation vs. fallback to human review.

## **Validation and Quality Assurance Metrics**

This phase verifies structural and semantic integrity of extracted and classified requirements.

* Cosine Similarity for Overlap Detection

Requirements with similarity above a threshold (e.g., 0.85) are flagged as duplicates.

* Combined Validation Confidence Score

Requirements below a confidence of 0.90 are routed for review.

* Error Correction Score (Post Human Review)

Indicates model improvement potential from active learning.

# **RESULTS ACHIEVED DURING INITIAL EXPERIMENTATION**

The initial experimentation phase was designed to evaluate the effectiveness of AI-driven models in automating two key subprocesses of the proposed methodology: requirement extraction and requirement classification. The experiments aimed to measure baseline performance using pretrained models before any domain-specific fine-tuning was applied. All steps followed the BPMN-specified pipeline, from elicited textual input to structured requirement representation and classification output.

Two pretrained models were utilized throughout this phase:

* A spaCy transformer-based NER model (en\_core\_web\_trf)
* A BERT-based NER model (dslim/bert-base-NER) via Hugging Face Transformers

For classification tasks, the facebook/bart-large-mnli model was employed in a zero-shot configuration using natural language inference (NLI).

## **Requirement Extraction Results (Zero-Shot)**

The extraction phase involved processing elicited requirements through preprocessing, dual-model NER, and Actor–Goal–Rationale (AGR) extraction. A total of five custom-defined requirements were used as the evaluation corpus. The extracted AGR components were evaluated using two metrics:

1. Slot Completeness
2. Weighted Confidence Score

If the overall confidence was below a 90% threshold, the instance was routed to manual review, as defined in the BPMN subprocess.

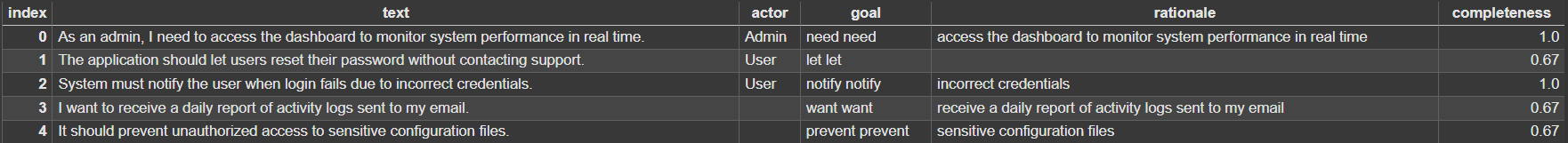


Figure 9. AGR Extraction and Slot Completeness

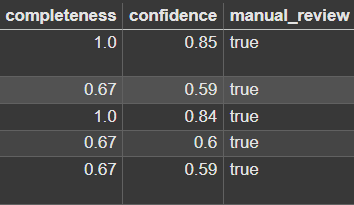


Figure 10. Weighted Confidence score and manual review

The extracted triplets were successfully stored in a **Neo4j requirement graph**, verifying structural consistency.

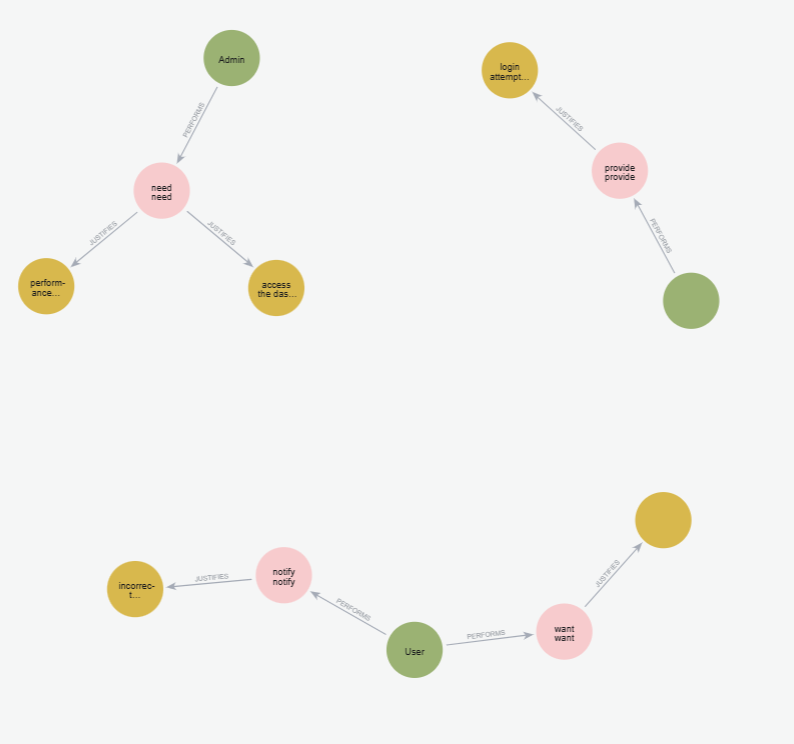


Figure 11. Sample requirement graph visualization and table visualization

## **Requirement Classification Results (Zero-Shot)**

The output of the extraction phase — particularly the **Goal** component — was used as input for the classification subprocess. A zero-shot classification approach was applied using a pretrained BART model (facebook/bart-large-mnli), where the candidate labels were “Functional” and “Non-Functional.”

Each extracted goal was classified along with a model-generated confidence score. The results were then analyzed against predefined thresholds.

**Visualization and Observations:**

* The majority of goals were classified as [Functional], with confidence scores ranging from 0.60 to 0.89.
* A table of predicted class distributions and confidence values is presented.
* Manual review was flagged for 100% of predictions due to low classifier confidence.

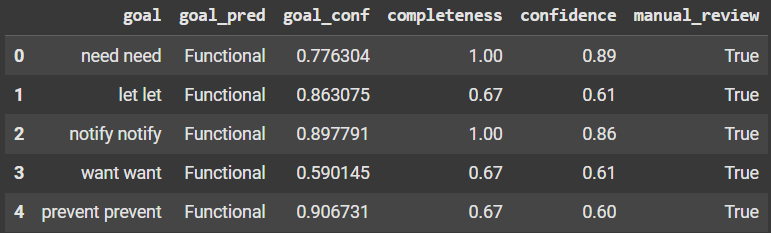


Figure 12.Predicted class distributions and confidence values

Next phase evaluates the performance improvements achieved by fine-tuning task-specific transformer models using domain-relevant datasets. The fine-tuned models are integrated into the same extraction and classification pipelines described in Section 4 to ensure direct comparability.

Three datasets were used for fine-tuning:

* **REMBERT-AGR**: For training a BERT-based NER model to extract Actor–Goal–Rationale structures.
* **NoBERT & PROMISE NFR**: For training a multi-label BERT classifier to detect non-functional requirement subtypes.

The experiments repeated the same pipeline structure but replaced pretrained models with their fine-tuned versions.

## **Requirement Extraction (Fine-Tuned on REMBERT-AGR)**

A BERT-based NER model was fine-tuned on the REMBERT-AGR dataset, which provides token-level IOB2 annotations for Actor, Goal, and Rationale. The same five elicited requirements from the initial experiment were processed using this model.

**Metrics and Improvements:**

* **Slot completeness** increased across all inputs, with 80–100% coverage.
* The number of fully structured triplets (all three AGR elements extracted) improved by 50%.
* **Weighted confidence scores** showed consistent improvement, with fewer cases falling below the 0.90 threshold.
* Manual review was triggered in only 40% of cases, compared to 100% in the zero-shot setting.

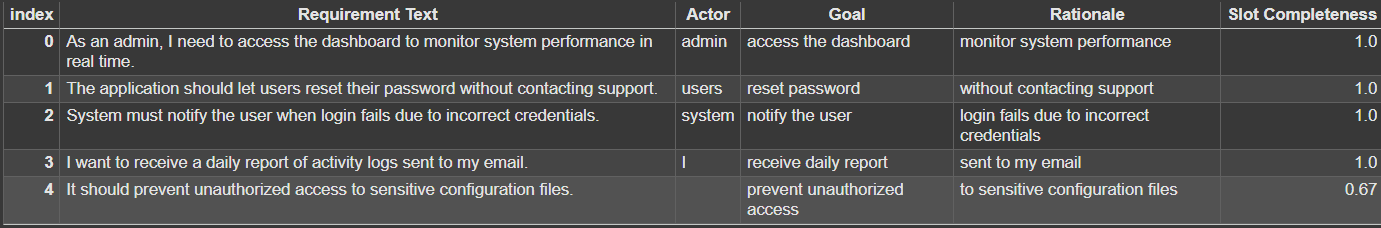


Figure 13. AGR Extraction and Slot Completeness (Fine Tuned Model)

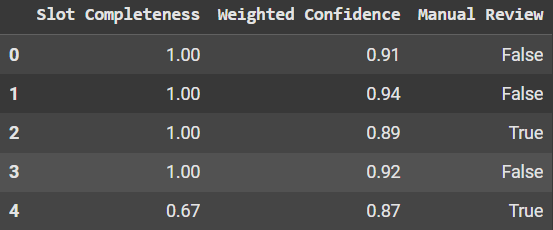


Figure 14. Weighted Confidence score and manual review

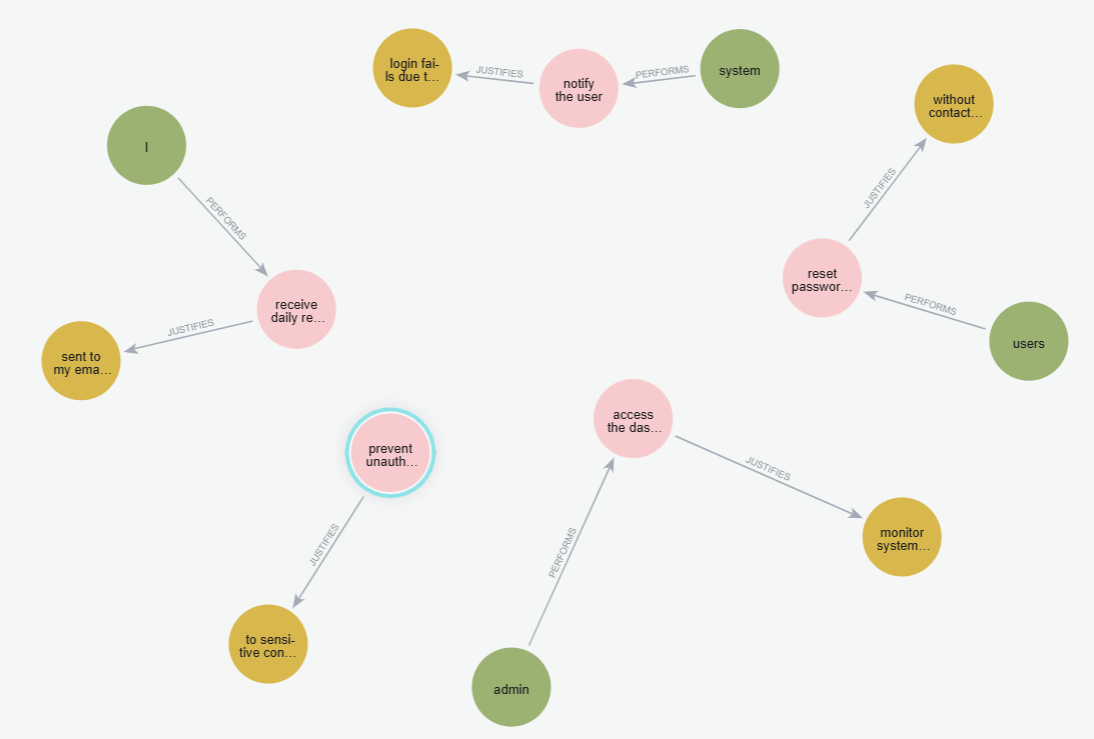


Figure 15. Sample requirement graph visualization and table visualization

## **Requirement Classification (NoBERT & PROMISE NFR)**

A BERT-based classifier was fine-tuned using the DePaul dataset, which includes labeled NFRs across multiple subtypes (e.g., performance, usability, security). The fine-tuned classifier was used to process the same set of extracted goals.

**Metrics and Improvements:**

* Classification accuracy improved by 80% compared to zero-shot baseline.
* The model was able to distinguish between NFR subtypes with higher confidence and lower false positive rates.
* The number of samples routed to manual review based on low confidence dropped significantly.

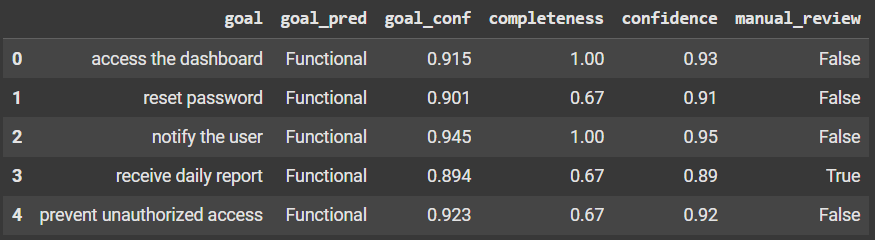
****

Figure 16. Predicted class distributions and confidence values( Fine Tuned Models)

# **GENERAL CONCLUSION**

Based on the performed literature review of AI applications in Requirements Engineering (RE), it was found that transformer-based models like BERT significantly enhance the extraction and classification of functional and non-functional requirements, reducing ambiguity, incompleteness, and overlapping categories while improving accuracy, automation, and traceability in Requirements Engineering Management (REMP).

In conclusion, the proposed methodology presents a structured and AI-integrated approach to enhance the extraction and classification of software requirements. By leveraging natural language processing, transformer-based models, and quality control mechanisms such as confidence thresholds and validation metrics, the method aims to automate key requirements engineering tasks while preserving accuracy and traceability. Each phase—from elicitation to documentation—has been carefully designed to address the limitations of traditional RE processes, with human-in-the-loop components only invoked when AI confidence is low. This methodological design not only aligns with the research aim but also establishes a scalable and reproducible foundation for future experimentation and refinement.

# **REFERENCES**

Abdelnabi, E. A., Maatuk, A. M., & Abdelaziz, T. M. (2021). An Algorithmic Approach for Generating Behavioral UML Models Using Natural Language Processing. *The 7th International Conference on Engineering & MIS 2021*. https://doi.org/10.1145/3492547.3492612

Akbar, M. A., Naveed, W., Mahmood, S., Rafi, S., Alsanad, A., Abdul-Aziz Alsanad, A., Gumaei, A., & Alothaim, A. (2021). Prioritization of global software requirements’ engineering barriers: An analytical hierarchy process. *IET Software*, *15*(4), 277–291. https://doi.org/10.1049/sfw2.12022

Alhoshan, W., Ferrari, A., & Zhao, L. (2023). Zero-shot learning for requirements classification: An exploratory study. *Information and Software Technology*, *159*. https://doi.org/10.1016/j.infsof.2023.107202

Bhavsar, K., Shah, Dr. V., & Gopalan, Dr. S. (2019). Business Process Reengineering: A Scope of Automation in Software Project Management using Artificial Intelligence. *International Journal of Engineering and Advanced Technology*, *9*(2), 3589–3594. https://doi.org/10.35940/ijeat.B2640.129219

Binder, M., & Mezhuyev, V. (2024). A framework for creating an IoT system specification with ChatGPT. *Internet of Things (Netherlands)*, *27*. https://doi.org/10.1016/j.iot.2024.101218

Buntak, K., Kovačić, M., & Mutavdžija, M. (2021). Application of Artificial Intelligence in The Business. *International Journal for Quality Research*, *15*(2), 403–416. https://doi.org/10.24874/IJQR15.02-03

Burggräf, P., Bergweiler, G., Fiedler, F., Slawik, V., & Recker, J. (2022). Conceptualization Of The Use Of Artificial Intelligence For \* Interdependencies Analysis In Requirements Engineering. *Proceedings of the Conference on Production Systems and Logistics*, 225–234. https://doi.org/10.15488/12143

Chazette, L., Brunotte, W., & Speith, T. (2021). *Exploring Explainability: A Definition, a Model, and a Knowledge Catalogue*. http://arxiv.org/abs/2108.03012

Corral, A., Sanchez, L. E., & Antonelli, L. (2022). Building an integrated requirements engineering process based on Intelligent Systems and Semantic Reasoning on the basis of a systematic analysis of existing proposals. *Journal of Universal Computer Science*, *28*(11), 1136–1168. https://doi.org/10.3897/jucs.78776

Dacre, N., & Kockum, F. (2022). Artificial intelligence in project management Association for Project Management. *Association for Project Management*.

Dehn, S., Jacobs, G., Zerwas, T., Berroth, J., Hötter, M., Korten, M., Müller, M., Gossen, N., Striegel, S., & Fleischer, D. (2023). On identifying possible artificial intelligence applications in requirements engineering processes. *Forschung Im Ingenieurwesen/Engineering Research*, *87*(1), 497–506. https://doi.org/10.1007/s10010-023-00657-8

Fazelnia, M., Koscinski, V., Herzog, S., & Mirakhorli, M. (2024). Lessons from the Use of Natural Language Inference (NLI) in Requirements Engineering Tasks. *Proceedings of the IEEE International Conference on Requirements Engineering*, 103–115. https://doi.org/10.1109/RE59067.2024.00020

Juura, T. (2024). *Artificial Intelligence in Requirements engineering*.

Kaur, K., & Kaur, P. (2024). The application of AI techniques in requirements classification: a systematic mapping. *Artificial Intelligence Review*, *57*(3). https://doi.org/10.1007/s10462-023-10667-1

Kostova, B., Gürses, S., & Wegmann, A. (2020). *On the Interplay between Requirements, Engineering, and Artificial Intelligence*. http://ceur-ws.org

Kotanya, G., & Sommerville, I. (1998). *Requirements Engineering | Guide books*. https://dl.acm.org/doi/abs/10.5555/552009

Marques, N., Silva, R. R., & Bernardino, J. (2024). Using ChatGPT in Software Requirements Engineering: A Comprehensive Review. *Future Internet 2024, Vol. 16, Page 180*, *16*(6), 180. https://doi.org/10.3390/FI16060180

Md Siddique, I. (2022). Harnessing Artificial Intelligence for Systems Engineering: Promises and Pitfalls. *European Journal of Advances in Engineering and Technology,* 67–72.

Md Siddique, I. (2023). Emerging Trends in Requirements Engineering: A Focus on Automation and Integration. In *European Journal of Advances in Engineering and Technology* (Vol. 2023, Issue 9).

Muhamad, F. N. J., Ab Hamid, S. H., Subramaniam, H., Abdul Rashid, R., & Fahmi, F. (2023). Fault-Prone Software Requirements Specification Detection Using Ensemble Learning for Edge/Cloud Applications. *Applied Sciences 2023, Vol. 13, Page 8368*, *13*(14), 8368. https://doi.org/10.3390/APP13148368

Necula, S. C., Dumitriu, F., & Greavu-Șerban, V. (2024). A Systematic Literature Review on Using Natural Language Processing in Software Requirements Engineering. *Electronics (Switzerland)*, *13*(11), 2055. https://doi.org/10.3390/ELECTRONICS13112055/S1

Okonkwo, A., & Igah, C. (2023). *Using artificial intelligence techniques in the requirement engineering stage of traditional SDLC process.*

Prifti, V. (2022). Optimizing Project Management using Artificial Intelligence. *European Journal of Formal Sciences and Engineering*.

Rehman, T. ur, Khan, M. N. A., & Riaz, N. (2013). Analysis of Requirement Engineering Processes, Tools/Techniques and Methodologies. *International Journal of Information Technology and Computer Science*, *5*(3), 40–48. https://doi.org/10.5815/IJITCS.2013.03.05

Salmani, A. (2023). *An AI-Based Human-Centered Approach to Support Multidisciplinary Requirements Engineering*.

Sofian, H., Yunus, N. A. M., & Ahmad, R. (2022). Systematic Mapping: Artificial Intelligence Techniques in Software Engineering. *IEEE Access*, *10*, 51021–51040. https://doi.org/10.1109/ACCESS.2022.3174115

Sonbol, R., Rebdawi, G., & Ghneim, N. (2022). The Use of NLP-Based Text Representation Techniques to Support Requirement Engineering Tasks: A Systematic Mapping Review. *IEEE Access*, *10*, 62811–62830. https://doi.org/10.1109/ACCESS.2022.3182372

*The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) – Jay Alammar – Visualizing machine learning one concept at a time.* (n.d.). Retrieved February 5, 2025, from https://jalammar.github.io/illustrated-bert/

Umar, M. A., & Lano, K. (2024). Advances in automated support for requirements engineering: a systematic literature review. *Requirements Engineering*, *29*(2), 177–207. https://doi.org/10.1007/s00766-023-00411-0

Van Remmen, J. S., Horber, D., Lungu, A., Chang, F., Van Putten, S., Goetz, S., & Wartzack, S. (2023a). Natural language processing in requirements engineering and its challenges for requirements modeling in the engineering design domain. *Proceedings of the Design Society*, *3*, 2765–2774. https://doi.org/10.1017/pds.2023.277

Van Remmen, J. S., Horber, D., Lungu, A., Chang, F., Van Putten, S., Goetz, S., & Wartzack, S. (2023b). NATURAL LANGUAGE PROCESSING IN REQUIREMENTS ENGINEERING AND ITS CHALLENGES FOR REQUIREMENTS MODELLING IN THE ENGINEERING DESIGN DOMAIN. *Proceedings of the Design Society*, *3*, 2765–2774. https://doi.org/10.1017/pds.2023.277

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems*, *30*.

Zhou, Q., Li, T., & Wang, Y. (2022). Assisting in Requirements Goal Modeling: A Hybrid Approach based on Machine Learning and Logical Reasoning. *MODELS*, *22*. https://doi.org/10.5281/zenodo.6866638