

سحاب الحميد

Sahab_Alhumad_Thesis_(2)_AI901.docx

- 1- Final Thesis (Word Format)
- Thesis_(2)_AI901_Dr_Khaled_Eskaf_Third Trimester 2025 (Moodle PP)
- Midocean University

Document Details

Submission ID

trn:oid::1:3450707876

Submission Date

Dec 25, 2025, 11:15 PM GMT+4

Download Date

Dec 26, 2025, 12:51 AM GMT+4

File Name

xcod.304053297_201621_109IA__2__sisehT_deamuhIA_bahaS_سحاب_الحميد_83792

File Size

735.9 KB

38 Pages

6,996 Words

45,103 Characters

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?

The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.





An AI-Driven Framework for Gathering and Documenting Business Requirements

Prepared By

Name: [Sahab Alhumaed] **IDNo:**[25100261]

**Submitted in fulfillment of the requirements for the
degree Master of Science in Artificial Intelligence**

At

College of Informatics

Midocean University

UAE

Supervisor Name: Dr. Khaled Eskaf

**College of Informatics, Midocean University,
Comoros**

[2025-Semester Three]

Acknowledgements

I am grateful to my supervisor Dr. Khaled Eskaf for his guidance, support, and ongoing encouragement in my preparation of this thesis. His valuable feedback and insightful comments greatly enriched and enhanced this work.

I would also like to express my gratitude to all who have helped me, whether directly or indirectly, in the completion of this thesis, in terms of continued support and assistance.

Last but not least, I must say how grateful I am to my family for their constant encouragement, patience, and unwavering support throughout my studies. My thesis has been shaped, due in large part, by the confidence they have in me.

Abstract

Software requirements are central to software development success, but getting them clear and classifying them accurately is still a major challenge. In this thesis, we propose an AI-Proposed Framework to automatically classify Functional Requirements (FRs) and Non-Functional Requirements (NFRs) and to detect ambiguity within requirement statements through machine learning and natural language processing techniques.

The framework is trained and evaluated using the PROMISE_exp dataset of 969 labeled requirements. Several machine learning models were evaluated, including Support Vector Machines (SVM), Logistic Regression, Naïve Bayes, and Random Forest. SVM achieved the best performance, with an accuracy of 90.7% for binary FR/NFR classification and a macro-F1 score of 0.74 across eleven NFR subcategories. It also includes a rule-based ambiguity detection module that is able to recognize vague adjectives, passive voice constructions, weak modal verbs and other linguistic patterns that lessen the clarity of requirements.

The findings show that automatic ambiguity detection can greatly improve early stage software development with higher requirement accuracy and lower level of misunderstanding.

The framework also generates Business Requirements Documents (BRDs) automatically in order to facilitate real-world use, so that stakeholders will have a clear and interpretable summary of requirements classifications and quality assessments. Combined, these elements create the AI-Proposed Framework, a complete AI pipeline to map requirement categorization, ambiguity detection, and automated BRD generation. Possible future work, among others, may be to expand the data, tune the ambiguity detection rule set, and investigate deep-learning methods to improve classification accuracy.

Table of Contents

.....	1
An AI-Driven Framework for Gathering and Documenting Business Requirements	1
Acknowledgements	2
Abstract	3
<i>List of Abbreviations.....</i>	<i>6</i>
<i>List of Figures.....</i>	<i>6</i>
<i>List of Tables.....</i>	<i>6</i>
<i>Chapter 1 Introduction</i>	<i>7</i>
1.1 Problem Definition	7
1.2 Thesis Objectives	7
1.3 Thesis Contribution.....	8
1.4 Thesis Organization.....	9
<i>Chapter 2 Literature Review</i>	<i>10</i>
2.1 Introduction of Automated Requirements Analysis.....	10
2.2 Functional and Non-Functional Requirements.....	10
2.3 Natural Language Processing for Requirements Engineering.....	11
2.4 Classical Machine Learning Techniques for Requirement Classification.....	11
2.5 Deep Learning Approaches in Requirement Engineering	12
2.6 Quantitative Results Summary	12
2.7 Research Gaps and Opportunities	13
2.8 Conclusion	13
<i>Chapter 3 Related Work.....</i>	<i>15</i>
3.1 Introduction	15
3.2 Functional vs. Non-Functional Requirement Classifications	15
3.3 Multi-Class Classification of Non-Functional Requirements	16
3.4 Ambiguity Detection in Software Requirements	17
3.5 Datasets and Difficulties for Evaluation in Previous Work.....	17
3.6 Comparative Analysis and Identification of Limitations.....	18
3.7 Summary and Justification of the Proposed Framework.....	19
<i>Chapter 4 Proposed Approach</i>	<i>20</i>
4.1 Data Collection and Description.....	20
4.2 Data Preprocessing	20

4.3 Feature Engineering	21
4.4 Feature Selection.....	21
4.5 Model Selection and Optimization	22
4.6 Ambiguity Detection.....	22
4.6.1 Duplicate Requirement Handling and Traceability	22
4.7 BRD Generation	23
4.7.1 BRD Automation Workflow	23
4.8 Summary	24
<i>Chapter 5 Experimental Results and Discussion</i>	<i>25</i>
5.1 Experimental Setup	25
5.2 Baseline Model Performance	26
5.3 Evaluation of Multi-Class Performance	27
5.4 Feature Importance Analysis.....	30
5.5 Results of the Ambiguity Detection.....	32
5.6 Comparison to Related Research	32
5.7 Discussion	34
5.8 Summary	35
<i>Chapter 6 Conclusion and Future Work</i>	<i>36</i>
6.1 Conclusion	36
6.2 Future Work	37
<i>References</i>	<i>38</i>

List of Abbreviations

- **AI:** Artificial Intelligence
- **BoW:** Bag of Words
- **BRD:** Business Requirements Document
- **CNN:** Convolutional Neural Network
- **F1-score:** Harmonic Mean of Precision and Recall
- **FR:** Functional Requirement
- **LR:** Logistic Regression
- **LSTM:** Long Short-Term Memory
- **ML:** Machine Learning
- **NB:** Naïve Bayes
- **NFR:** Non-Functional Requirement
- **NLP:** Natural Language Processing
- **PROMISE_exp:** PROMISE Expanded Requirements Dataset
- **RF:** Random Forest
- **RNN:** Recurrent Neural Network
- **SMOTE:** Synthetic Minority Over-sampling Technique
- **SVM:** Support Vector Machine
- **TF-IDF:** Term Frequency–Inverse Document Frequency

List of Figures

Figure 1. Thesis organization and logical flow of chapters in the proposed research	9
Figure 2. Overview of the proposed methodology illustrating the end-to-end workflow from requirement input to automated BRD generation.	20
Figure3. Binary classification performance comparison of baseline models.	26
Figure4. Confusion matrix for binary FR/NFR classification	27
Figure5. Macro-F1 score comparison of multi-class classifiers	28
Figure6. Distribution of predicted NFR categories (confusion matrix).....	29
Figure 7. Top 20 influential features for binary FR/NFR classification.....	30
Figure 8. Influential terms for each NFR category.	31
Figure 9. Comparison of the proposed framework performance with related studies.....	33

List of Tables

Table 2.1 Quantitative Performance Summary of Related Studies	13
Table 2 Summary of Related Work	33

Chapter 1 Introduction

1.1 Problem Definition

Software requirements describe what a system should do and the characteristics it should possess. Analysts traditionally manually categorized these requirements into Functional Requirements (FRs) and Non-Functional Requirements (NFRs) by hand a lengthy process prone to personal bias and susceptible to ambiguity.

While NFRs (including performance, security, usability, and reliability) are paramount to building a high-quality system, they are often poorly identified or poorly defined.

The situation is compounded by the ambiguity in natural-language requirements. A number of requirements use ambiguous adjectives or sentences like “fast” or “user-friendly,” use weak modal verbs “should” and “may” or lack measurable criteria.

This type of ambiguity is the source of a lot of confusion in teams, causes additional costs, rework, and project delay. For these reasons, an automated method of accurately classifying requirements and identifying ambiguity in the software development life cycle is needed.

In this thesis, the term software requirements refers to system-level requirements expressed in natural language, as provided in the PROMISE_exp dataset. The Business Requirements Document (BRD) generated by the proposed approach does not represent raw business-level requirements. Instead, it serves as a structured, stakeholder-oriented analytical summary derived from software requirements, consolidating requirement classification outcomes, ambiguity detection results, and quality assessments.

1.2 Thesis Objectives

In this thesis, I will propose an AI-Proposed Framework to collect and summarize requirements for the business. The primary aims are:

1. Develop a binary classification model to identify the Functional Requirements and Non-Functional Requirements.
2. Train a multi-classifier to classify NFRs into one of eleven categories: security, performance, usability, availability, and operability.
3. Rule-based linguistic analysis can detect ambiguity in requirements statements. It also helps to track down vague, incomplete, or unclear expressions.

4. Create structured Business Requirements Documents (BRDs) that combine classification results and findings on ambiguity, as well as suggestions for improvements.
5. Assess classification model performance by using standard metrics (accuracy, precision, recall, F1-score), and contrast them to outcomes of the literature.

1.3 Thesis Contribution

This work provides the following key contributions:

- **Integrated Analysis Pipeline:** It presents AI-driven requirements gathering, a system that automates requirement classification, ambiguity detection, and documentation generation within a single workflow.
- **High-Performance Classification Models:** The research creates machine learning models, especially SVM classifiers, that work well. For example, they have 90.7% accuracy in FR/NFR classification and a macro-F1 score of 74% across eleven NFR categories.
- **Quality and Ambiguity Assessment:** A linguistic module based on rules can find seven main types of ambiguity. This lets the system sort requirements and check how clear and complete they are.
- **Stakeholder-Oriented BRD Generation:** The framework makes Business Requirements Documents (BRDs) on its own. These documents list the types of requirements, point out any unclear points, and make suggestions. This helps people talk to each other on project teams.
- **Scalability and Extensibility:** The methodology is built to be improved in the future, such as by adding transformer-based models, expanding NFR categories, and taking user feedback into account.
- **Academic and Practical Value:** The thesis links old ways of doing requirements engineering with new ways of using AI. It gives researchers a way to repeat their work and software development teams a useful tool to help them make decisions.

1.4 Thesis Organization

This thesis consists of six chapters that outline the research background, methods, and findings. With its main focus being introducing the problem and objectives, the paper goes on to highlight literature on similar topics and current work; then it goes into more detail with these principles. Finally, the proposed AI-driven framework is introduced, and practical results from experiments on it are done to test these principles. The thesis finishes by summarising contributions and directions for future research. The organization of the thesis and flow of thoughts is depicted in Figure 1.

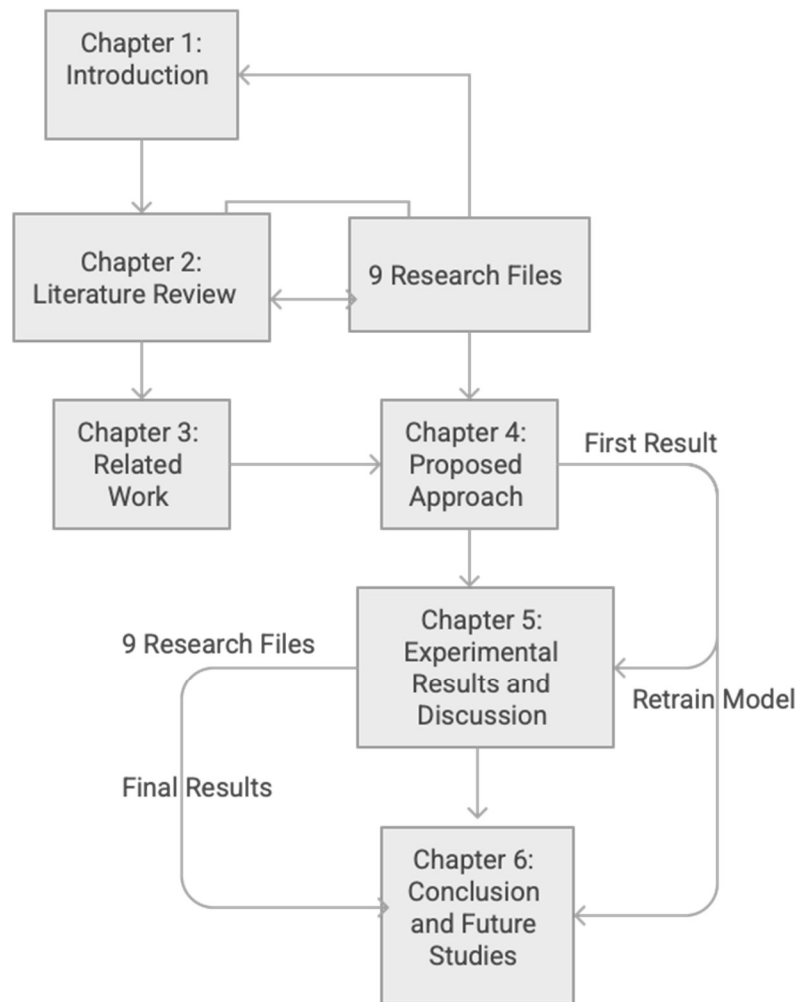


Figure 1. Thesis organization and logical flow of chapters in the proposed research

Chapter 2 Literature Review

2.1 Introduction of Automated Requirements Analysis

Software requirements engineering is a critical aspect of software development projects, as requirements define both system functionality and quality attributes. Requirements give us information regarding what a system needs to be able to do as well as the quality attributes it needs to fulfill. Requirements are written for the benefit of all stakeholders often in natural language, but as we all know, the wording cannot always be the same depending on the source and also need be interpreted differently. In the past few decades, as software systems have grown more intricate and numerous, the manual definition and categorization of requirements are time-consuming and prone to errors.

Automatic requirement analysis is therefore an important area of research focused on assisting the goals toward more efficient, consistent and higher quality requirements engineering.

ML-related automatic systems using NLP can automatically classify the requirements, find the quality issues and be used to catch the defects early during the development process. These approaches reduce human bias while improving scalability and standardization across projects.

In this chapter we will elaborate on a discussion of the conceptual foundations and technical toolkits for automated requirements analysis of the proposed AI framework.

2.2 Functional and Non-Functional Requirements

There are two general types of software requirements: Functional Requirements (FRs) and Non-Functional Requirements (NFRs). FRs define the functionality of the system, also what kind of services, interactions, or actions that must occur (data processing or user actions).

The quality attributes of NFRs govern how the system works such as performance, security, usability, availability, and reliability.

FRs are generally explicit and action-oriented while NFRs are typically implicit and abstract, and they can be expressed qualitatively. In this way, we see NFRs are harder to identify and classify, especially when they are included in functional statements. Inaccurate NFR detection will cause problems for us.

Therefore we end up with systems that remain fully functional but perform poorly in practical applications.

2.3 Natural Language Processing for Requirements Engineering

The technical base of the analysis of the textual requirements comes from NLP and these data are passed into natural language processing. Unstructured requirement statements are used for these analyses; NLP structures are a framework to turn them into formats that are suitable for machine learning language algorithms.

Typical NLP methods for preprocessing NLP steps include tokenization, lowercasing, stop-word removal, punctuation removal, and normalization of textual variations. Feature representation approaches, such as Bag of Words (BoW), Term Frequency–Inverse Document Frequency (TF-IDF), and n-grams, are commonly used in requirement engineering applications.

We adopt word and short phrase importance schemes within requirement statements and can learn by number linguistic structures of the requirement type. NLP methods also facilitate ambiguity perception through linguistic patterns including vague adjectives, weak modal verbs, ambiguous pronouns and passive voice constructions, which often lead to unclear requirements.

2.4 Classical Machine Learning Techniques for Requirement Classification

Classical machine learning algorithms have historically been used to solve requirement classification problems and are generally regarded as being well-suited for small to medium-sized datasets. SVMs are preferred because of their capacity to support high-dimensional and sparse feature spaces generated by text vectorization methods such as TF-IDF.

Another regularly used linear classifier, Logistic Regression, is easily interpretable and can be trained in an effective manner but is likely to encounter difficulties with complex linguistic features.

Naïve Bayes classifiers are an efficient and implementable system; however, they frequently fall short of their potential in natural language text since they assume feature independence, and performance in terms of accuracy may suffer. Random Forest models enable ensemble-based learning and feature importance, but may be less efficient on data which is sparse.

These traditional models still offer powerful baselines and practical applications in automatic requirement categorization, especially when datasets are limited in size.

2.5 Deep Learning Approaches in Requirement Engineering

Deep Learning techniques have recently been studied to enhance requirement classification performance. Convolutional Neural Networks (CNNs) have been employed to learn the local linguistic structure. Likewise, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) classifiers can infer sequential dependencies of requirement statements. Models based on transformers like BERT incorporate contextual embeddings and self-attention, which promotes a deeper understanding of the semantic layers. However, such models normally need large annotated datasets and computational resources.

Datasets from requirement engineering are often small and imbalanced, a serious problem in practical applications, therefore limiting the usefulness of deep learning-based approaches. Therefore, although deep learning holds promising potential, classical machine learning models are still more feasible as techniques for many real-world requirement analysis tasks. Recent research in 2024 also highlights that semantic overlap among NFR categories remains the primary source of misclassification, rather than deficiencies in learning algorithms.

This observation suggests that improvements in ambiguity handling and requirement quality assessment are as important as advances in classification models.

2.6 Quantitative Results Summary

Several studies have reported quantitative results for automated requirement classification using machine learning techniques. Support Vector Machines combined with TF-IDF features consistently demonstrate strong performance for FR/NFR classification. However, multi-class NFR classification remains challenging due to class imbalance and semantic overlap among categories.

Table 2.1 summarises representative quantitative results reported in prior research and provides context for the performance achieved by the proposed framework.

Table 2.1 Quantitative Performance Summary of Related Studies

Study	Dataset	Features	Algorithm	Accuracy	F1-score
Rahman et al. (2023)	PROMISE_exp	TF-IDF	Linear SVM	–	81.5%
Binkhonain (2023)	PROMISE_exp	BoW	Naïve Bayes	63.0%	–
AI-Proposed Framework (This Thesis)	PROMISE_exp	TF-IDF (bi-grams) + SMOTE	SVM	90.7%	74% (macro)

2.7 Research Gaps and Opportunities

Recent studies published in 2024 and 2025 further confirm that classical machine learning models, particularly SVM combined with TF-IDF features, remain competitive for requirements classification tasks when dataset size is limited. Although transformer-based models have shown promise, their effectiveness is often constrained by the scarcity and imbalance of publicly available requirements datasets. These findings support the continued relevance of classical ML approaches in practical requirements engineering settings.

Despite the good results achieved in the studies before it, there were still some gaps in automated requirement engineering. Many previous efforts target standalone tasks like FR/NFR classification or NFR category prediction with no integrated pipelines.

As we continue to struggle with dataset size and imbalance, model generalization is limited, while overlapping NFR semantics introduce classification ambiguity.

In addition, automatic recognition of linguistic ambiguity and writing more structured documents, such as Business Requirements Documents (BRDs), are rarely mentioned. These gaps indicate that there is a major need for all-inclusive AI-Proposed Frameworks that offer classification, quality assessment, and documentation assistance. In addition, there are hardly any researches about duplicate requirements arising even among multiple source documents, and not many of them offer automated methods to combine such requirements without losing their traceability. This limitation is especially critical for real-world industrial environments.

2.8 Conclusion

The idea of automated requirement analysis had a deep conceptual and technical approach with requirement types, NLP techniques, machine learning models, and existing challenges covered in this chapter.

Although previous work has proved that machine learning-based approaches (such as SVM-based techniques) performed effectively, issues still exist concerning integration, ambiguity handling, and documentation generation.

These gaps call upon the AI-Proposed Framework to overcome them as an integrated AI-driven pipeline. The following chapter reviews related work in detail and then compares existing approaches with the proposed solution.

Chapter 3 Related Work

3.1 Introduction

The research literature on automated software requirement analysis has expanded greatly over the past few years due to the development of more sophisticated software systems and the need to integrate these requirements based on natural language requirements.

Traditional requirements engineering requires a lot of manual input from domain experts, and this has been found to be tedious, subjective and inconsistent, especially for large-scale projects. Accordingly, in recent years scientists have turned more attention to Machine Learning (ML) and natural language processing (NLP) approaches to automate requirement classification and quality assurance processes.

This chapter reviews the relevant empirical works that deal with

- (1) Functional Requirements and Non-Functional Requirements separation (FRs and NFRs).
- (2) Differentiating from Functional Requirements and Non-Functional Requirements (NFRs) in the scope of multi-class classification of the model.
- (3) Recognizing ambiguity of the requirement statements.

In contrast to Chapter 2, which provided theoretical rationale and methodology, this chapter describes how existing research solved problems, explored the issue, and compared the solutions, as well as the advantages and disadvantages of earlier solutions. These studies inform the design choices taken in the AI-Proposed Framework.

3.2 Functional vs. Non-Functional Requirement Classifications

One of the first and most prominent problems of automated requirements engineering is the concept of both FR vs. NFR classification of requirements.

Kurtanović and Maalej (2017) were one of the first to focus on this issue, and it is shown that supervised machine learning models like Support Vector Machines (SVMs) can produce great accuracy and recall when trained on lexical features obtained from requirements text.

Their findings indicated that SVMs could provide a rich linguistic patterning even when only a small amount of data was available. These observations have bolstered these results in more recent studies.

Rahman et al. (2023) assessed 15 machine learning algorithms with the PROMISE_exp dataset and found that linear SVM combined with TF-IDF features

had an F1-score of 81.5% for FR/NFR classification. Their work highlighted the need to represent the variables correctly and demonstrated that simpler linear models often perform better than more complex classifiers with smaller datasets.

Binkhonain (2023) examined probabilistic classifiers especially Naïve Bayes and observed relatively inferior accuracy (63%) with the same dataset. This project demonstrated a frequently seen weakness of Naïve Bayes classifiers, namely their strong assumption of feature independence which is seldom met in natural language.

The application of linear SVM models was reinforced in more recent research in 2024, as they were able to perform well over FR/NFR classification, particularly when the techniques were coupled with robust preprocessing and feature engineering. These works stress that basic models tend to generalize better than complex deep learning architectures when training data is limited and heterogeneous.

Together, these works demonstrate that SVM-based methods are better than the rest for FR/NFR classification tasks.

3.3 Multi-Class Classification of Non-Functional Requirements

A broader classification process can be achieved by multi-class classification of non-functional requirements. Apart from binary classification, a number of studies have explored the more difficult task of separating them into specific quality attributes e.g., security, usability, performance, and reliability.

Muhamad et al. (2022) investigated this task using the PURE dataset and reported that SMOTE-based balancing improved SVM classification performance (78.7% accuracy and 78.9% F1-score), highlighting the impact of class imbalance in NFR datasets.

Multi-Class NFR Classification is intrinsically much harder, as there is a semantic overlap between categories, according to other authors. For instance, requirements related to performance frequently use words that are similar to terms of operability or reliability requirements, resulting in frequent misclassification.

Kurtanović and Maalej (2017) also noted this problem, and found that although overall precision remained high, confusion related to closely related NFR categories remained with the most efficient features. These results support the notion that whilst the classification framework of NFRs by machine learning models, even the interpretation of overlapping semantics and uneven distributions for class spaces by

machine learning models are challenges that need to be confronted, which in turns might be motivated by the need for more advanced preprocessing, feature engineering and evaluation methods.

3.4 Ambiguity Detection in Software Requirements

Ambiguity detection is less studied in the literature as compared to requirement classification. Abd El Sameaa et al. (2021) noted that ambiguity of requirements, including ambiguous adjectives, weak modal verbs, and the lack of quantitative constraints in requirements presents as a major hindrance in agile software development environments.

Their research stressed that ambiguous requirements often complicate the situation by misinterpretation, causing more rework and resulting in delayed projects. Most methods for ambiguity detection are based on rule-based linguistic methods, since machine learning-based models need annotated ambiguity datasets in high volumes and are challenging to generate.

These rule-based systems typically address known ambiguity patterns (e.g., passive voice constructions, vague pronouns). Although effective, such approaches are typically deployed as standalone methods, without integration into existing requirement classification pipelines. However, integrating ambiguity detection within automated requirement classification pipelines remains limited in the literature, despite the strong relationship between requirement quality and requirement type in practice.

Despite renewed interest in ambiguity detection in 2024 and 2025, most proposed solutions still operate independently from requirement classification systems. Integrated frameworks that jointly address classification, ambiguity detection, and stakeholder-oriented documentation remain scarce.

3.5 Datasets and Difficulties for Evaluation in Previous Work

A widespread constraint encountered by most prior studies is that publicly available requirements datasets are often small and imbalanced. For example, PROMISE_exp and PURE contain limited labeled samples for some NFR categories, which increases the difficulty of multi-class evaluation and reduces generalization.

This drawback severely limits the use of deep learning models to the point where researchers often must turn to classical machine learning methods. The labeling

inconsistency and writing style variance among the projects also add noise which reduces model generalization.

Other studies have demonstrated evaluation inconsistencies, such as reporting accuracy measures without equivalent precision, recall, and F1-score, which presents challenges in cross-study comparison. These difficulties necessitate robust preprocessing along with balanced evaluation mechanisms and standardized reporting, all of which are explored in the design of the proposed framework.

3.6 Comparative Analysis and Identification of Limitations

Despite notable progress in automated requirements engineering, a comparative analysis of existing studies reveals several persistent limitations. First, many approaches address individual tasks in isolation, such as FR/NFR classification or NFR categorization, without providing an integrated end-to-end pipeline. In particular, requirement quality checks, including ambiguity detection, are often treated as standalone problems or are entirely omitted from classification workflows.

Second, stakeholder usability remains limited in most prior work. Most studies have largely been limited to evaluation metrics (e.g., accuracy and F1-score) reporting, rarely producing a structured, practitioner-oriented output (e.g., Business Requirements Documents (BRDs)) that can be directly used at work. That means that most proposed solutions have limited applicability for practice. Third, dataset-related challenges exist in the majority of studies such as small size of data, class imbalance, and semantic overlap between closely related NFR categories. These issues negatively affect model generalization and frequently lead to misclassification between similar quality attributes.

Finally, traceability and duplicate handling across multiple source documents are rarely addressed, despite being common challenges in real-world industrial environments where requirements are collected from heterogeneous artifacts. These limitations highlight the need for unified frameworks that combine requirement classification, ambiguity analysis, duplicate handling, and automated documentation within a single, coherent pipeline.

3.7 Summary and Justification of the Proposed Framework

In this section, a review of the studies that focused on automated requirement classification, multi-class NFR prediction, ambiguity detection, and data problems were presented.

Although previous studies show that machine learning, especially SVM models, could perform well for requirement classification, they have also found great gaps, such as broken pipelines, a lack of ambiguity handling, and no automated documentation support.

These discrepancies directly contribute towards the creation of the AI based framework, a fully-fledged AI- driven system that combines requirement classification, ambiguity detection, and Business Requirements Document generation in one process.

The detailed discussion of the proposed method is provided in the following chapter: the architecture, preprocessing methods, models, and elements addressing the issues encountered in this chapter.

In conclusion, previous works exhibit that machine learning has been successful in requirements classification, but provide insufficient pipelines that combine classification, ambiguity detection, duplicate handling as well as automated documentation generation. This AI-based solution is intended to fill these gaps by bringing a complete end-to-end solution that more accurately represents requirements engineering approach in the real world.

Chapter 4 Proposed Approach

In this chapter, we describe how we constructed the AI-Proposed Framework, an AI-embedded system that classifies software requirements and evaluates their quality. We will use this methodology to build a robust, automated workflow that provides Functional vs. Non-Functional Requirement identification (FR vs. NFR), NFR class classification system, ambiguity detection capability, and Business Requirements Document (BRD) generation for the end-user.

The methodology includes multiple important stages (dataset preparation, preprocessing, feature engineering, model training, quality analysis, etc.). These steps improve performance, interpretability, and practical usefulness to software engineering teams.



Figure 2. Overview of the proposed methodology illustrating the end-to-end workflow from requirement input to automated BRD generation.

4.1 Data Collection and Description

Dataset Consistency Statement. All experiments reported in this thesis are conducted using a single dataset, namely the PROMISE Expanded Requirements Dataset (PROMISE_exp), which contains 969 labeled software requirements. All tests of this work are with a single dataset, PROMISE Expanded Requirements Dataset (PROMISE_exp), that comprises 969 labeled software requirements. This guarantees that the identical dataset is utilized for all the tasks: binary FR/NFR classification, multi-class NFR classification into eleven NFR classes, and ambiguity analysis. In all measurements the reproducibility and comparability of the results are increased by employing the same preprocessing pipeline, and the same 80/20 train–test split configuration.

4.2 Data Preprocessing

These preprocessing steps ensure that the learning algorithms process clean and consistent textual inputs suitable for feature extraction and classification.

Preprocessing of the text was performed to correctly classify the requirements. SMOTE was implemented to address the imbalance with respect to classes in TF-IDF feature space, and to counter the majority-class bias to stabilize classification when minority groups are underrepresented.

- Tokenization: Requirement statements can be segmented into distinct tokens.
- Lowercasing: The words are all presented in lowercase to minimize inconsistency.
- Stop-word and punctuation deletion: Words and symbols that do not contribute information (not informative) are removed.
- Diacritic and accent normalization: characters are all normalized as far as possible from non-uniform representations.

4.3 Feature Engineering

We apply the Term Frequency–Inverse Document Frequency (TF-IDF) vectorization method to transform the textual requirements into numerical representations that are readily interpretable in a machine learning setting.

TF-IDF highlights critical terms while giving low weights to common but less meaningful terms.

Bigrams are added to capture context beyond a single word. Such short word sequences offer richer semantic information, helping the model better differentiate between requirement categories. In this way, feature engineering provides high dimensional vectors suitable for understanding requirement content without losing some meaningful contextual information.

4.4 Feature Selection

Explicit feature selection was not applied to preserve linguistic interpretability. Using TF-IDF with bigrams provides informative weighting of terms, and linear SVM inherently performs implicit feature weighting, allowing discriminative terms to influence classification decisions without removing features.

4.5 Model Selection and Optimization

Several machine learning models were tested:

- Support Vector Machines (SVM).
- Logistic Regression.
- Naïve Bayes.
- Random Forest.

The SVM classifier did better than classical models with an accuracy of 90.7%.

(1) Binary Classification (FR vs. NFR)

The SVM classifier achieved better performance than our other models overall with an accuracy of 90.7%. It outperformed all models. Its advantage is in dealing with high-dimensional, sparse TF-IDF features and creating effective decision boundaries.

(2) Multi-Class Classification: NFR Category Prediction

An eleven-class classifier was trained on NFRs. Multi-class SVM also had the best performance, achieving a macro-F1 score of 0.74, which means that it has been successfully effective even in the case of diverse and overlapping NFR subtypes. These results confirm the robustness of SVM techniques against high-dimensional text classification challenges, and the applicability of SVM-based methods for requirement engineering tasks.

4.6 Ambiguity Detection

4.6.1 Duplicate Requirement Handling and Traceability

In practice, the same requirement can also come up in multiple source documents like SRS (Software Requirements Specification) and meeting notes. The framework contains a duplicate-detection step prior to generating summary statistics and documentation to prevent double counting while ensuring traceability.

Requirement normalization is performed in three steps: lowercasing, punctuation removal, whitespace normalization. TF-IDF vectors represent the normalized requirements, and cosine similarity is computed for pairwise similarity.

Together, requirements with a similarity larger than a specified threshold are combined into a single canonical requirement.

Each canonical requirement is counted once in all aggregate statistics in the generated Business Requirements Document (BRD). While keeping references to all originating documents/identifiers where duplicated requirements are mentioned in the documentation of the organization, traceability is maintained.

The following approach creates reporting accuracy, avoids redundancy, and maintains auditability across multiple sources of requirements.

4.7 BRD Generation

The final component of the AI-Proposed Framework (AI-PF) generates a Business Requirements Document (BRD) that provides a structured, stakeholder-oriented summary of all analysis results. The BRD includes the following elements:

- Overall counts of Functional Requirements (FRs) and Non-Functional Requirements (NFRs)
- Distribution of NFR subcategories
- Percentage of clear versus ambiguous requirements
- Lists of classified FRs and NFRs
- Detailed ambiguity findings with suggested revisions
- Project overview and consolidated requirement summaries

4.7.1 BRD Automation Workflow

The Business Requirements Document (BRD) is generated automatically from the outputs of the analysis pipeline. For each canonical requirement, the framework stores the following information: (1) the original requirement text, (2) the predicted requirement type (FR or NFR), (3) the predicted NFR category where applicable, (4) detected ambiguity flags and associated linguistic patterns, (5) suggested revision guidance, and (6) traceability metadata, including references to all source documents in which the requirement appeared.

Using this structured information, the framework automatically produces stakeholder-oriented summaries, including overall requirement counts, distributions of NFR categories, ambiguity rates, and detailed tables of classified requirements. Finally, the BRD is programmatically exported using a template-based generator (e.g., DOCX, PDF, or Markdown), ensuring consistent formatting, reusability, and repeatability across different projects.

4.8 Summary

This chapter provided details of the AI-Proposed Framework methodology. It explained the dataset properties, preprocessing procedures, feature engineering, classification approaches, ambiguity detection rules, and the BRD generation procedure. Together, these ingredients create a pipeline for automated software requirement characterization. The experimental setup and evaluation are covered in the next chapter for the performance assessment of the described scheme.

Chapter 5 Experimental Results and Discussion

In this chapter, we present the experimental results for the development and testing of the AI based requirement analysis framework. Various machine learning paradigms have demonstrated good results for binary FR/NFR classification, multi-class classification model, feature contribution analysis, and ambiguity detection on real-life software requirement datasets.

5.1 Experimental Setup

The full experiment is executed in Python using widely used machine learning libraries including Scikit-learn, Pandas, NumPy and Matplotlib. For the primary dataset, PROMISE_exp is composed of labeled Functional Requirements (FRs), and Non-Functional Requirements (NFRs), the latter scattered over eleven NFR categories. A number of preprocessing steps were performed before fitting the model:

- **Normalization:** Lowercasing, punctuation removal, and stop-word filtering.
- **Tokenization:** A method of splitting requirement statements into tokens.
- **Diacritic normalization** for textual inconsistencies.
- **TF-IDF vectorization** by means of bigrams to capture short semantic patterns.
- **Balancing by SMOTE** for binary and multi-class problems to address class imbalance.
- **80/20 train–test split** to guarantee the unbiased evaluation.

The models evaluated were the following:

- Support Vector Machine (SVM).
- Logistic Regression.
- Naïve Bayes.
- Random Forest.

The binary classifier separated FRs from NFRs, and the multi-classifier predicted one of eleven subcategories of NFRs.

5.2 Baseline Model Performance

The first series of experiments analysed each model using the identical configuration of TF-IDF + bigram feature.

Binary FR/NFR Classification

- Among all the models tested, the SVM classifier showed the best performance with:
 - Accuracy: 90.7%.
 - Constant high precision, recall and F1-score.
- In contrast:
 - Due to the assumption of feature independence which is not a realistic premise for natural-language text, Naïve Bayes performed poorly.
 - Logistic Regression and Random Forest also performed moderately, but fell short of the stable and accurate results of SVM under high-dimensional representations.
- **Figure 3** compares accuracy, precision, recall and the F1-score of the baseline models.

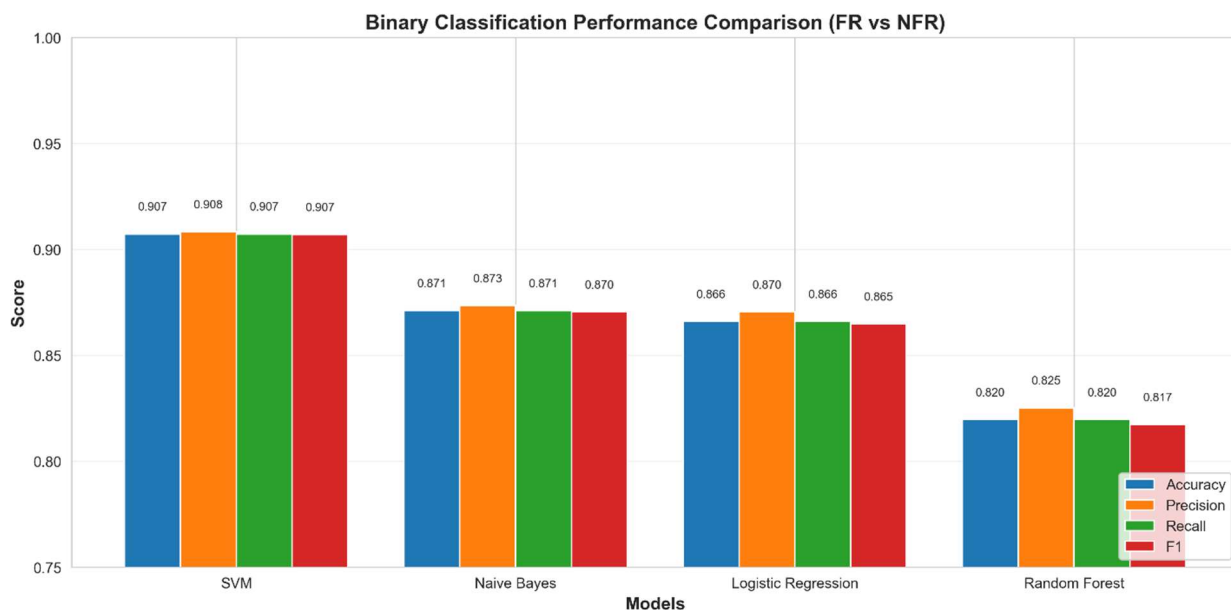


Figure3. Binary classification performance comparison of baseline models.

- The confusion matrix is shown in **Figure 4** to indicate the pattern of correct vs. incorrect FR/NFR categories as well as the common misclassification behaviours.

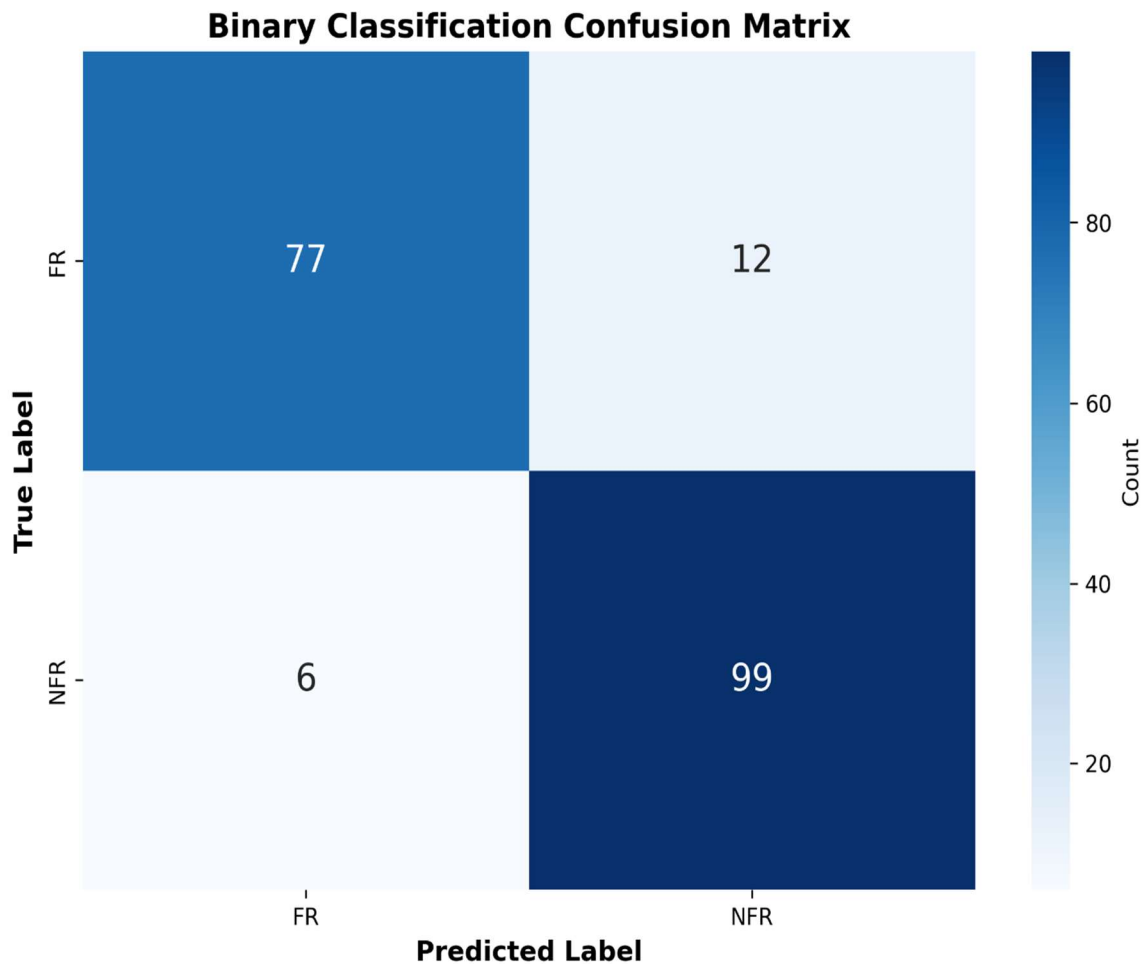


Figure4. Confusion matrix for binary FR/NFR classification

5.3 Evaluation of Multi-Class Performance

- The multi-class task was designed to evaluate the performance of each model in classifying NFRs into eleven categories.
- The strong performance was shown again of the SVM classifier:
 - Macro-F1 Score: 0.74.
- Such performance demonstrates the robustness of the model despite several challenges of NFR classification:

- Small sample sizes in some NFR categories.
- Overlapping semantics among related categories (e.g., performance vs. operability).
- High linguistic variability in natural-language requirement statements.
- **Figure 5** compares the macro-F1 performance of the different models to confirm the superiority of SVM.

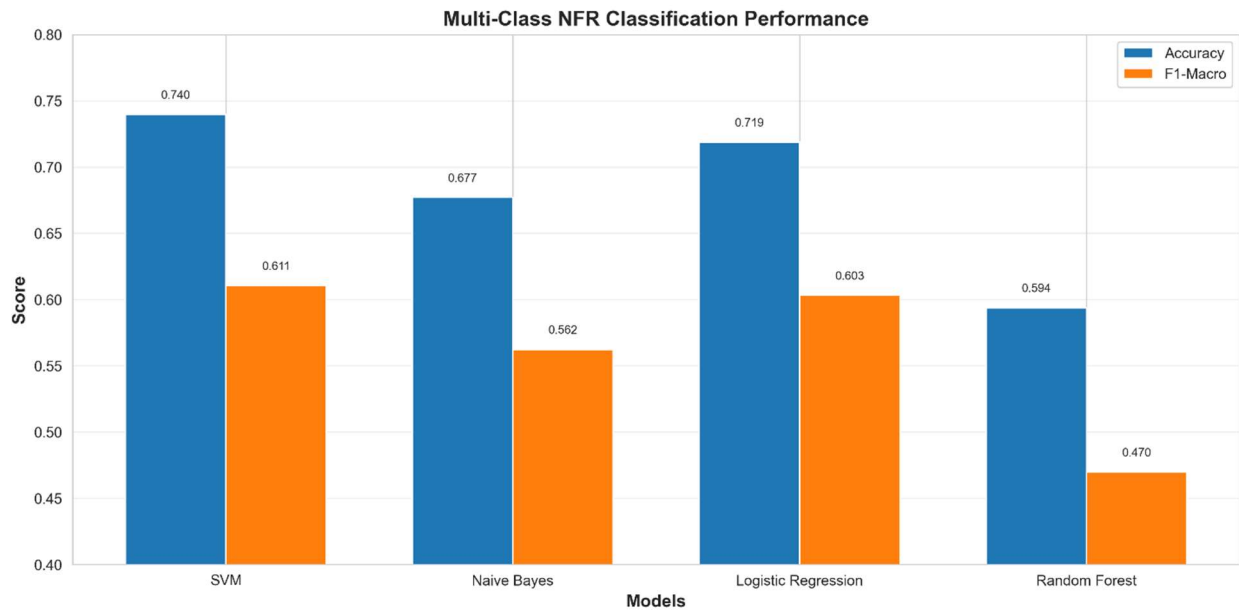


Figure5. Macro-F1 score comparison of multi-class classifiers

- The distribution of the predicted NFR categories, both predicted correctly and where confusion occurred, is shown in **Figure 6**.

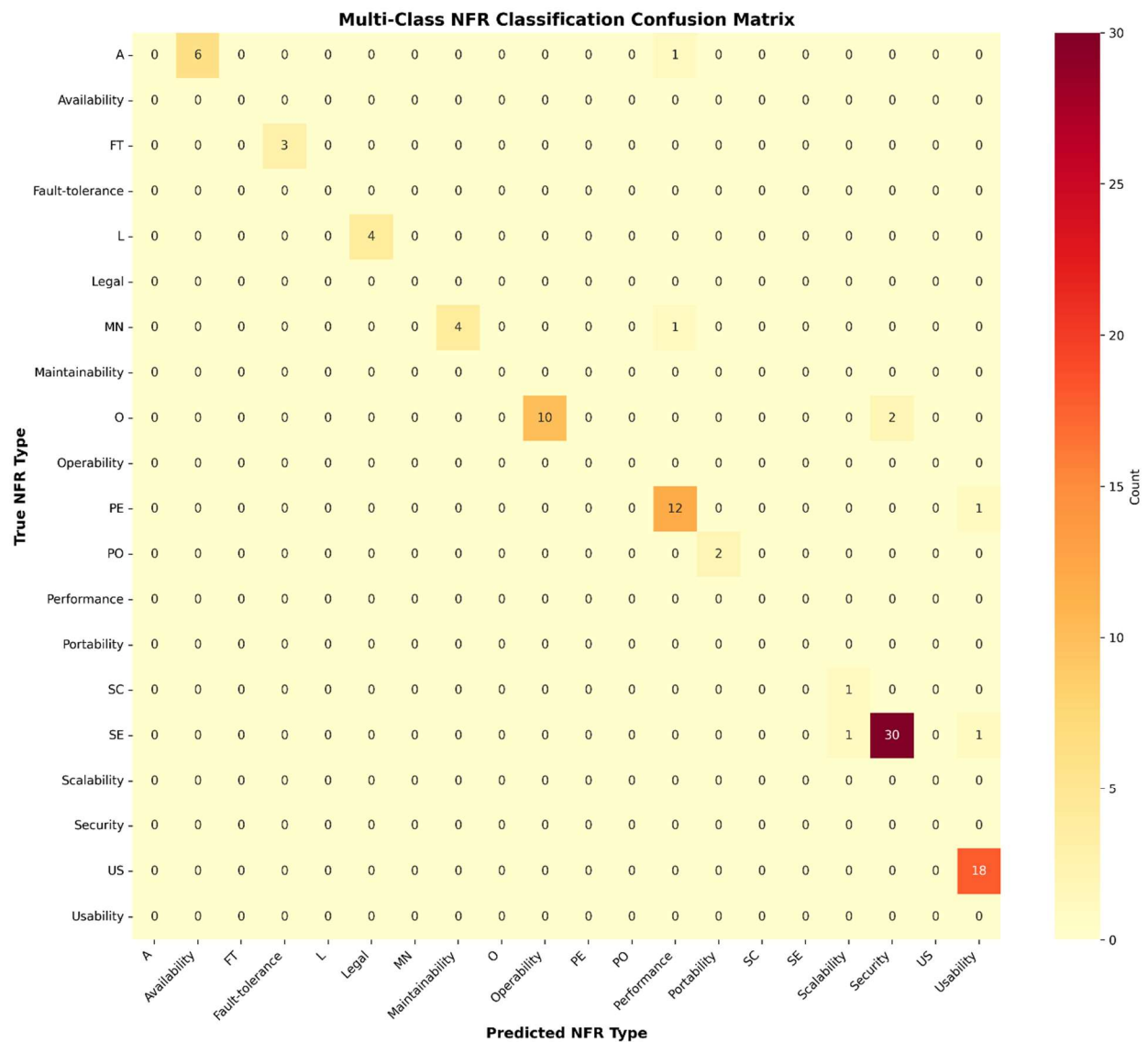


Figure6. Distribution of predicted NFR categories (confusion matrix)

5.4 Feature Importance Analysis

Feature importance was examined for binary and multi-class tasks to further understand the decisions the model makes.

(1) Binary FR/NFR Classification

- Figure 7 shows the top 20 most influential terms.
- Examples include:
 - **NFR-indicative:** server, seconds, access
 - **FR-indicative:** allow, display

(2) Multi-Class NFR Classification

- **Figure 7** highlights keywords strongly associated with specific NFR types:

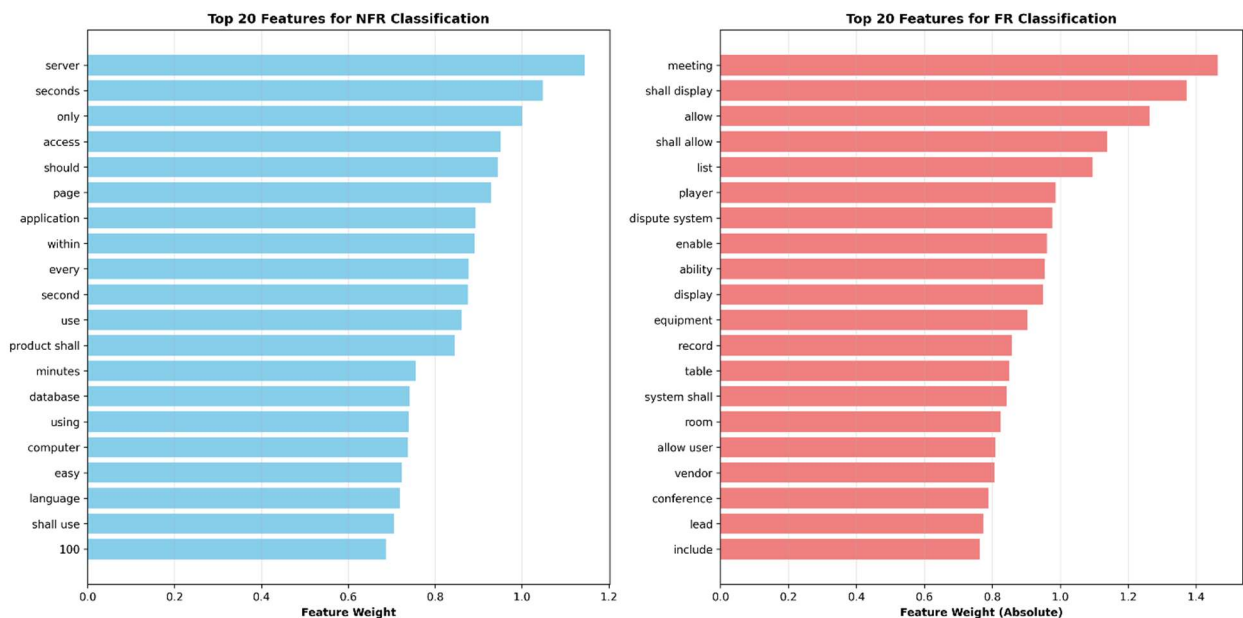


Figure 7. Top 20 influential features for binary FR/NFR classification.

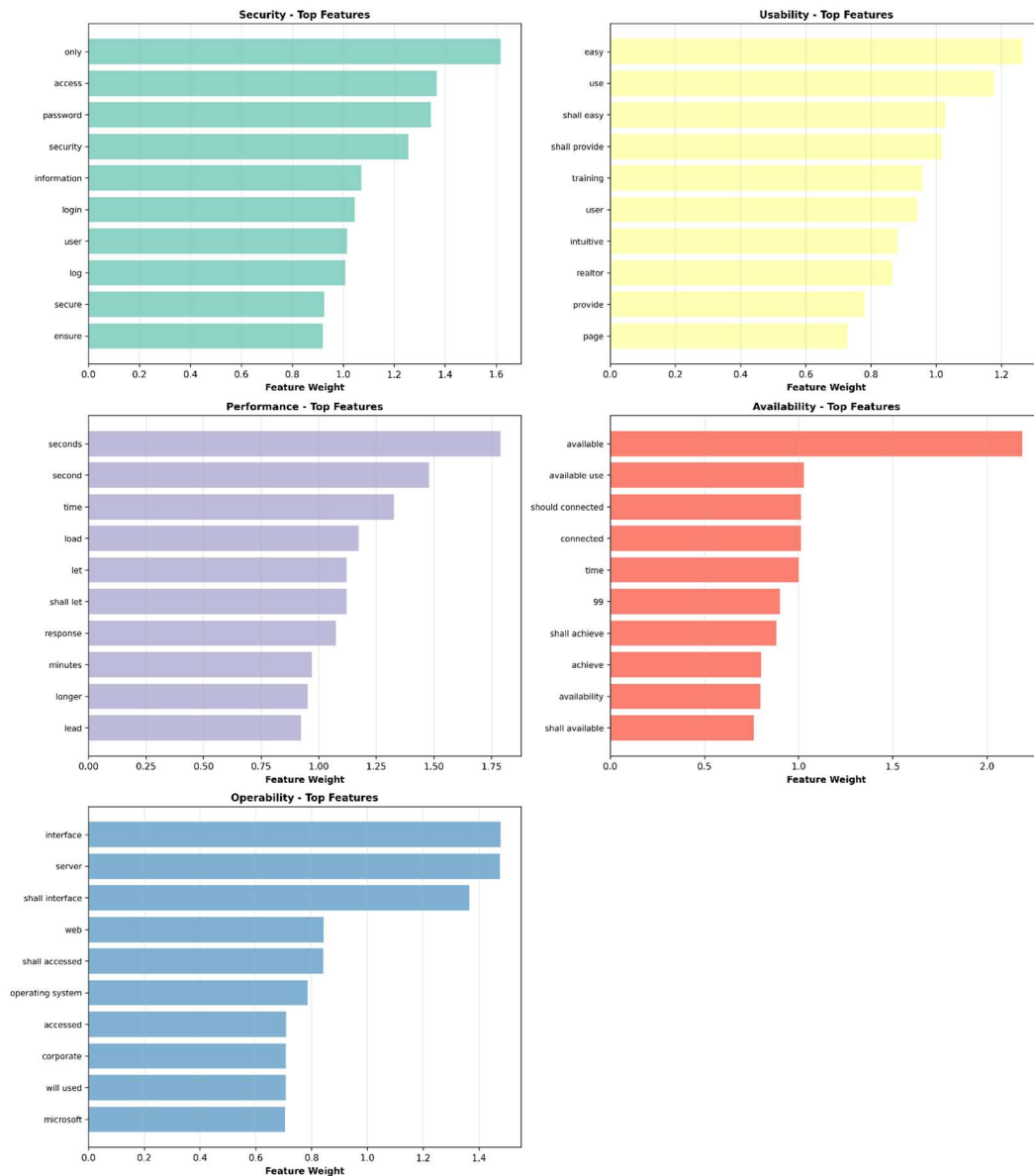


Figure 8. Influential terms for each NFR category.

- **Security:** password, access
- **Usability:** easy, use
- These findings illustrate how certain domain-specific terms shape the distinction between requirement classes and categories.

5.5 Results of the Ambiguity Detection

The rule-based ambiguity detection module discovered ambiguous phrasing in about 95% of the requirements with ambiguity markers. Examples include:

- **Vague adjectives:**
 - “fast response time” → Suggested change: specify clear quantitative thresholds.
- **Weak modal verbs:**
 - should, may → Indicate as non-committal.
- **Ambiguous pronouns:**
 - it, they → Highlighted for unclear referents.
- **Passive voice:**
 - Marked when responsibility wasn't clear.

These findings highlight the value of linguistic quality assessment, and also demonstrate the potential for the system to help understand more clearly the requirements specifications.

Quantitative benchmarking for ambiguity detection was not performed due to the limited availability of publicly annotated ambiguity datasets at scale. Therefore, the ambiguity module is evaluated qualitatively through detected patterns and illustrative examples.

5.6 Comparison to Related Research

For context on the performance of the proposed framework, we compare the SVM classifier in Figure 9 with previous works.

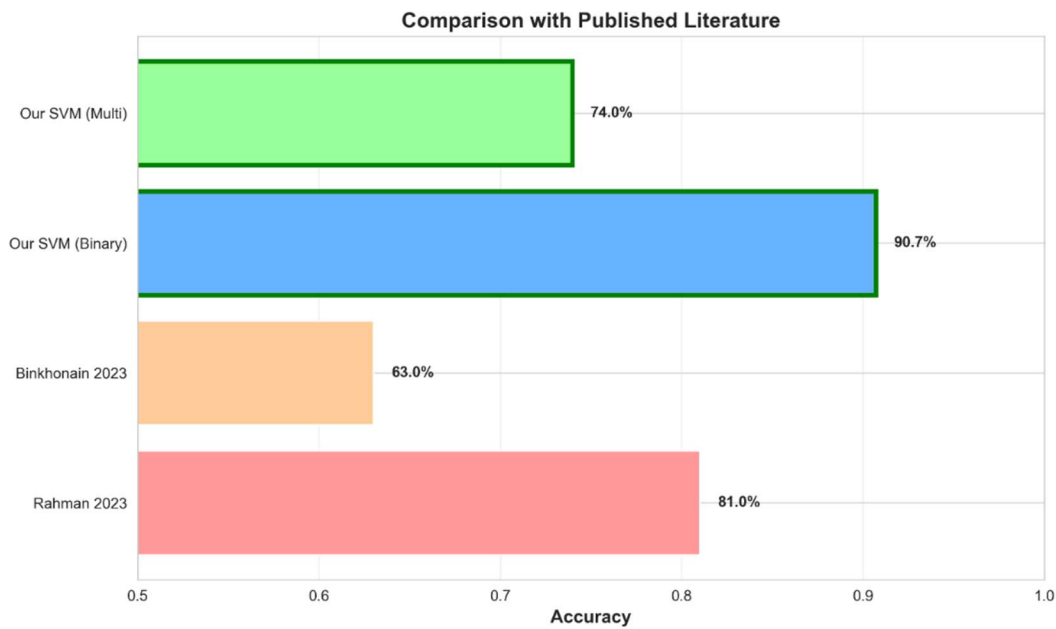


Figure 9. Comparison of the proposed framework performance with related studies.

Key Observations:

- Our binary SVM classifier has better performance than the existing data (90.7% accuracy and 91% F1 score), Binkhonain (2023)– 63% accuracy (Naïve Bayes), Rahman et al. (2023) — 81.5% F1 (Linear SVM).
- Performance improvements are attributed to class balancing (SMOTE) and richer textual representations using TF-IDF with bigrams.
- Multi-class Performance (macro-F1 = 0.74) is competitive with or superior to prior methods.

Table 2 Summary of Related Work

Study	Dataset	Features	Algorithm	Accuracy	Precision	Recall	F1-score
AI-Proposed Framework (This Thesis)	PROMISE_exp	TF-IDF(bigrams)	SVM	90.7%	91.0%	91.0%	91.0%
Rahman et al. (2023)	PROMISE_exp	TF-IDF	Linear SVM	—	—	—	81.5%

These results highlight the proposed framework as a meaningful advancement in automated requirement classification research.

5.7 Discussion

Now with these experimental results established, we can validate the effectiveness of the proposed framework and offer a few key takeaways:

- **Key Strengths**
 - We have also demonstrated that SVM models outperformed all other methods, both in binary tasks and in multi-class tasks, consistently.
 - We showed significant improvement in predictive performance through preprocessing and class balancing.
 - Ambiguity detection provided qualitative depth not offered by common classification-only methods.
- **Remaining Challenges**
 - Overlapping semantics makes multi-class NFR classification challenging.
 - However, the dataset size restricts the potential of deep-learning models.
 - Subsequent work could focus on transformer-based models (e.g., BERT) and human-in-the-loop refinement for ambiguity rules.
 - These findings highlight the potential and limitations of current AI-based requirement analysis approaches in general.

Additional considerations are summarized as follows:

Most misclassifications occurred between semantically overlapping NFR categories, such as performance and operability, indicating that the errors are largely due to inherent ambiguity in requirement wording rather than model limitations.

SMOTE was applied to mitigate class imbalance in both binary and multi-class settings. Although originally designed for numerical data, its application to TF-IDF-based text representations has been shown to improve classifier stability when minority classes are underrepresented.

Hyperparameters were selected empirically, and final performance evaluation was conducted on a held-out test set without cross-validation.

5.8 Summary

This chapter presented the experimental evaluation of the proposed framework, covering binary and multi-class classification results, feature-importance analysis, ambiguity detection outcomes, and comparisons with related literature. The system demonstrated strong quantitative and qualitative performance. The next chapter provides the overall conclusion of this thesis and outlines directions for future research.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

This description of the proposed framework, an AI-based approach for automatic software requirements classification and quality evaluation, forms the basis of this thesis. By combining machine learning models with rule-based linguistic structural analysis, the system addresses three main challenges in requirements engineering: distinguishing between Functional and Non-Functional Requirements (FRs vs. NFRs), classifying Non-Functional Requirements into eleven commonly recognized categories, and identifying ambiguity in natural language requirement statements. The main findings of the study are:

- **Leading classification systems:**

The SVM (Support Vector Machine) classifier achieved high scores in all tasks: 90.7% FR/NFR classification accuracy down to the macro-F1 score of 0.74 across the eleven NFR subcategories. These results were better than many prior works made on similar datasets and with feature extraction and showed the strong performance of SVMs on high-dimensional text analysis.

- **Effective ambiguity detection module:**

The rule-based linguistic module could identify vague, incomplete, and ambiguous conditions, such as weak modal verbs, ambiguous pronouns, measurable constraints not included, and passive forms of voice. This feature facilitates the early tightening of demands, improving the quality of the specification.

- **Business Requirements Documentation is simple and effective:**

The proposed framework produces Business Requirements Documents that summarize classification results, ambiguity discovery, and quality insights. This creates clarity and improves communication between development and user teams with ease.

- **An AI pipeline of requirement engineering in one:**

Preprocessing, classification, ambiguity detection and BRD generation together show the potential of AI for the automation and standardization of requirement engineering, thus minimizing manual effort, subjectivity and bias. Our work shows the potential for AI (especially when combined with linguistic heuristics) to enrich the automation of part of requirement analysis process.

Overall, the results suggest that artificial intelligence can contribute substantially to the modernization of requirements engineering through improving analytical accuracy and creating more complete and consistent requirements documentation to enable the automated classification and provision of assurance for requirements.

6.2 Future Work

The results are promising, but there are several possibilities for extending and strengthening the framework. The following could be further explored to enhance its capacity:

(1) Enhanced and Diverse Datasets

For more robust and generalizable models, the software requirements should be added from various domains like healthcare, finance, embedded systems, and transportation in future studies. A larger dataset would allow the model to learn a greater variety of linguistic patterns, reducing overfitting to the PROMISE_exp dataset.

(2) Integration of Sophisticated Deep Learning Models:

Transformers, contextual embeddings (BERT / RoBERTa), and hybrid deep learning designs may provide a great contribution to semantic understanding, particularly within multi-class NFR classification. These approaches may serve in resolving the ambiguity between two closely related categories and can therefore get deeper contextual information.

(3) Integration with Real-World Workflows:

Incorporating the proposed framework into industry solutions like Jira, Confluence, Azure DevOps, or IBM DOORS helps to smooth the adoption of this tool in real requirements engineering environments. Continuous validation and real-time classification could be achieved if automated analyses were embedded into requirements management pipelines.

(4) Integrating the Human in the Loop to Refine Ambiguity:

By doing so, we hope the proposed framework can grow into a highly scalable and intelligent requirements engineering assistant, supporting both academic researchers and professional software development companies. As the discipline continues to transition toward more automated and AI-supported engineering practices, such systems are expected to be increasingly used to enhance requirements quality and accelerate project delivery.

References

1. D. De Martino and F. Palomba, "Classification and challenges of non-functional requirements in ML-enabled systems: A systematic literature review," *Empirical Software Engineering*, 2025.
2. M. Rahman, A. H. M. R. Karim, and K. Sakib, "Non-functional requirements classification using machine learning algorithms," in *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, 2023, pp. 1–8.
3. A. Muhamad, N. Kama, and R. Ahmad, "A class balancing methods comparison in software requirement classification using a Support Vector Machine," *Journal of Software Engineering and Applications*, vol. 15, no. 4, pp. 78–89, 2022.
4. Z. Kurtanović and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *Proceedings of the IEEE International Requirements Engineering Conference (RE)*, 2017, pp. 490–495.
5. A. Abd El Sameaa, M. A. El-Sayed, and H. M. Hassan, "Challenges of non-functional requirements extraction in agile software development using machine learning," *Journal of Systems and Software*, vol. 176, pp. 110–123, 2021.
6. S. Binkhonain, "Machine learning approaches for software requirement classification," *International Journal of Computer Science and Information Security*, vol. 21, no. 2, pp. 45–52, 2023.
7. G. Airlangga, A. F. Rahman, and H. Suryotrisongko, "Enhancing software requirements classification using machine learning and feature engineering techniques," *Software Quality Journal*, 2024.
8. K. Rahman, M. S. Islam, and K. Sakib, "A deep learning framework for non-functional requirement classification," *Scientific Reports*, vol. 14, 2024.
9. J. Abbas, A. Ahmed, and M. Alenezi, "Classification and comprehension of software functional and non-functional requirements using advanced NLP techniques," *Information and Software Technology*, 2024.
10. S. Bashir, R. Khan, and M. Iqbal, "Requirements ambiguity detection and explanation using hybrid NLP approaches," *Journal of Systems and Software*, 2025.