An Innovative Approach to Extracting and Classifying Non-Functional Requirements Using Machine Learning and NLP

Muhammad Shafiq[1,*], Waeal J. Obidallah[2], Mubarak Albathan[2] and Tahir Kamal[3]

[1] School of Computer Science, Shandong Xiehe University, Jinan, China

[2] College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia

[3] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, China

INFORMATION

Keywords:
Non-functional requirements (NFR)
machine learning (ML)
natural language processing (NLP)
requirement classification
word embedding vectorization

DOI: 10.23967/j.rimni.2026.10.75552

SCIPEDIA Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería

RIMNI Revista Internacional Métodos numéricos para cálculo y diseño en ingeniería CIMNE

An Innovative Approach to Extracting and Classifying Non-Functional Requirements Using Machine Learning and NLP

Muhammad Shafiq[1,*], **Waeal J. Obidallah**[2], **Mubarak Albathan**[2] **and Tahir Kamal**[3]

[1]School of Computer Science, Shandong Xiehe University, Jinan, China

[2]College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia

[3]School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, China

ABSTRACT

This study presents a hybrid automated framework based on a combination of machine learning (ML) and natural language processing (NLP) approaches for the automatic categorization and extraction of non-functional requirements (NFRs) from free-text software development documents. Using the PROMISE dataset, this framework systematically integrates semantic representation learning, deep feature extraction, and kernel-based classification to improve the performance of NFR classification. Unlike current CNN-based approaches with end-to-end softmax-based classification, our proposed method fundamentally decouples feature learning from decision making. The first approach is to use Word2Vec embeddings to capture semantic context, and then use Convolutional Neural Networks (CNNs) as high-level feature extractors. An Improved Support Vector Machine with a Radial Basis Function kernel (ISVM-RBF) is applied for final classification, enabling more discriminative decision boundaries to be drawn in the high-dimensional semantic feature space. We reveal a considerable performance improvement with the CNN–Word2Vec setup, achieving as high as a 90% precision, significantly outperforming standard ML classifiers. The study points to three main findings: (i) CNN-based feature extraction is an efficient approach for finding and classifying NFRs, (ii) the semantic representation provided by word embedding methods is clearly superior to other traditional methods used in NLP, and (iii) NLP preprocessing of text is crucial for enhancing classification accuracy. Finally, ISVM-RBF adapts kernel-based classification over features derived from CNN, which enhances the robustness of the model to semantic overlaps between NFR categories and alleviates challenges posed by potentially large textual datasets required to train such models. This hybrid CNN–ISVM-RBF design constitutes the methodological novelty of the proposed method and effectively distinguishes it from current state-of-the-art methods in the literature.

OPEN ACCESS

Received: 03/11/2025

Accepted: 22/01/2026

DOI
10.23967/j.rimni.2026.10.75552

Keywords:
Non-functional requirements (NFR)
machine learning (ML)
natural language processing (NLP)
requirement classification
word embedding vectorization

*Correspondence: Muhammad Shafiq (shafiqk786@hotmail.com). This is an article distributed under the terms of the Creative Commons BY-NC-SA license

Nomenclature

NLP	Natural language processing
NFRs	Non-functional requirements
CNNs	Convolutional Neural Networks
ISVM-RBF	Improved Support Vector Machine with Radial Basis Function
RE	Requirements Engineering
NLTK	Natural Language Toolkit
ISVM-RBF	Improved Support Vector Machine with Radial Basis Function
KNN	K-Nearest Neighbor

1 Introduction

In the field of software engineering, requirement articulation is a crucial process performed both individually by designers and collaboratively with engineering teams. Customer expectations, regulatory frameworks, and the expertise of the teams help formulate these requirements, which form the basis for defining system functions and success criteria. As projects mature, these requirements evolve to address new and existing challenges and complexities, thereby aiding in project control, milestone tracking, and stakeholder communication. This constitutes a major task in which Requirements Engineering (RE) plays a pivotal role, ensuring that requirement specifications are complete, consistent, and relevant [1]. RE categorizes requirements into two types: functional requirements, which describe what the system should do (i.e., the actions it must perform), and non-functional requirements (NFRs), which focus on the system's quality attributes that are crucial to ensuring its performance and reliability [2]. The Software Requirements Specification (SRS) is a document that outlines the treatment of system complexity and aids in interpersonal communication within the team [3].

Over the past decade, the significant importance of RE has been clearly demonstrated through its presence across the software development lifecycle and the development of improved techniques for distinguishing system requirements. Since requirements are inherently complex and may contain ambiguity, many researchers have sought to move beyond manual approaches by leveraging advanced technologies such as NLP and ML to automate the challenge of accurately and correctly analyzing and classifying requirements [4].

However, current NFR classification techniques are primarily based on individual ML classifiers or DL models, and are sometimes hybrid without clear justification for the hybridization or theoretical justification for the integration of components. As a result, they lack robustness and explainability in practical RE situations. Inspired by these limitations, this paper presents a new hybrid NFR classification framework that incorporates Word2Vec-based semantic representations, Convolutional Neural Network (CNN) feature abstraction, and an Improved Support Vector Machine with Radial Basis Function (ISVM-RBF) classifier. While most conventional CNN-based methods base their decisions on softmax layers that directly output class scores, our proposed framework uses CNNs as feature extractors and relies on ISVM-RBF for final classification, which is more appropriate for high-dimensional and low-sample textual data. Our improvement with respect to ISVM-RBF includes adaptive kernel scaling and a one-vs-all decision strategy, which results in higher reparability and lower sensitivity to imbalanced NFR distributions. This design decision is theoretically driven by the effective behavior of kernel-based learning in non-linearly separable spaces and empirically designed for NFR classification problems.

In this paper, we propose a new automated method for classifying NFRs based on detailed syntactic and semantic analysis combined with ML approaches. The research aims to enhance the understanding and classification of these NFRs by focusing on key NFR types such as Fault Tolerance and Security [5–7]. This study makes three main contributions: (i) a novel hybrid architecture based on deep semantic feature extraction and kernel-based classification, grounded in theory, (ii) an explicit design rationale that distinguishes the proposed framework from CNN-only and SVM-only approaches, and (iii) a comprehensive evaluation strategy for small and imbalanced NFR datasets. The goals include refining the construction of an NFR detection model, evaluating the performance of the model with accurate metrics, and applying innovative procedures and evidential data to significantly improve development.

2 Literature Review

The state-of-the-art methods for extracting and classifying NFRs have made significant progress due to recent breakthroughs in ML and NLP. For example, Wang et al. [8] introduced TASTA, a Spatial and Temporal Attention Network, which achieved significant performance gains in text-based classification tasks. This underscores the pivotal role of ML and NLP in (semi) automating text mining, where ML is emerging as a dominant method to effectively distinguish between functional (FR) and non-functional (NFR) requirements. Huang et al. [9] further advanced the field by implementing a large language model simulator based on cold-start recommendations, enhancing recommender capabilities and highlighting the power of NLP applications, even in complex scenarios like software requirement classification. Additionally, significant contributions have been made in clustering techniques for NLP tasks, such as Belief Shift Clustering [10]. This work showcased full-domain convolutional attention for cross-lingual font style transfer and demonstrated the versatility of NLP models across various text classification tasks [11]. The innovations in ML and NLP discussed here illustrate the potential of these approaches to enhance NFR classification, aiding in the distinction between NFRs and FRs, and paving the way for more advanced and efficient text classification systems.

Based on syntactic and lexical features, a supervised ML approach was developed by 12. Kurtanovic and Maalej [12] which used two supervised ML algorithms, namely SVM and NB, to classify both FRs and NFRs with high recall rates [12,13]. Functional Requirements (FRs) and NFRs are two types of software requirements that should be effectively extracted during the requirements engineering process. Zhang et al. [10] expanded on this work with an empirical examination of SVM classifiers on NLP indices, and an evaluation of short NFR sentences, showing individual word indexes as the best predictors [14,15]. Semantic analysis and NFR classification play an important role in vectorization methods. In another study, Amasaki and Leelaprute [16] reviewed multiple vectorization methods, finding that Doc2Vec and SCDV outperformed other methods due to the complex nature of accurately identifying NFRs. Tóth and Vidács [17] added an additional time factor alongside precision and recall, and based on these measures, NB was found to be the most suitable classifier. Although existing studies demonstrate the effectiveness of classical ML models and representation spaces, they primarily use shallow semantic features and do not focus on deep feature abstraction and hybrid decision strategies.

Li et al. [18] introduced a cost-sensitive hierarchical classification model for large-scale datasets, which is crucial for tasks like NFR classification. Yin et al. [19] developed DPAL-BERT, a faster and lighter question-answering model that outperforms traditional models. Zheng et al. [20] further refined question answering with PAL-BERT, improving overall NLP task performance. Cao et al. [21] explored the dimensional needs of non-constructive learning, providing new insights for ML applications. Meng et al. [22] applied pre-trained language models to electric power audit text classification,

showcasing a domain-specific NLP application. Huang et al. [23] introduced EdgeLLM, a CPU-FPGA heterogeneous accelerator that improves the efficiency of large language models. Shen et al. [24] proposed VLCIM, a vision-language model for industrial defect detection, demonstrating the intersection of computer vision and NLP. Wu et al. [25] presented TFGIN, a graph inference network for table-based fact verification, showcasing the power of graph-based models in factual verification. Eli et al. [26] reviewed non-Latin natural scene text detection techniques, contributing to the expansion of NLP to multilingual environments. Wu et al. [27] also applied structure sampling and clustering for point cloud quality assessment, highlighting NLP's potential for data analysis. Liu et al. [28] developed SEAD-MGFE-Net, improving sentiment analysis through adaptive dropout and feature enhancement.

Cao et al. [29] conducted a comprehensive survey on low-resource data learning and further explored techniques for effectively learning from sparse datasets. For instance, Yin and Wang [30] used contrastive learning to distinguish whether the text was generated by AI or a human, which is another example showing that NLP contributes to science. Huang et al. [31] explored compiler tuning for inlining with ML, intersection software optimization and ML. Liu et al. [32] applied ML to optimize hydrodynamic shapes in underwater vehicles, utilizing ML for mechanical design. Wang et al. [33] introduced HHG-Bot, a graph-based model to detect Twitter bots, demonstrating the power of using graph models in social systems. Wang et al. [34] showed how hierarchical semantic extraction and semantic models can influence cybersecurity, improving Android malware detection. Wang et al. [35] focused on geometric matching for cross-modal retrieval, aimed at retrieval systems for different modalities of data.

Jiang et al. [36] investigated multi-label classification with a fuzzy similarity function, improving performance by considering semantic factors as part of the classification process. Ramadhani et al. [37] placed more importance on semantic aspects and proposed a system with high accuracy for classifying NFRs. While such semantic-driven approaches achieve greater classification accuracy, they often rely on handcrafted rules or similarity measures, resulting in limited scalability and an inability to cater to varying requirements and styles. In recent years, there has been some interest in applying Convolutional Neural Networks (CNNs) for NLP tasks, including requirement classification, showing that CNNs can achieve high precision and recall rates [38,39]. However, the challenge remains: how do the neural networks learn? Dekhtyar and Fong [40] showed improvements in classification accuracy with CNNs compared to traditional methods. However, despite their success, CNN-based NFR classification often uses end-to-end architectures and softmax classifiers, which are likely to overfit small and imbalanced datasets and facilitate low interpretability of learned decision boundaries. While research on RNN applications for NFR classification is limited, it has shown very promising results in text classification, with RNNs performing better than other models in terms of precision and recall [41]. Even though progress has been made in the automatic classification of NFRs, this task is still far from solved due to two main problems: firstly, a lack of properly labeled NFR datasets, and secondly, the inconsistency of NFR definitions in the literature. Casamayor et al. [42] proposed semi-supervised approaches to reduce labeling effort by utilizing non-categorized requirements. Ontology-based approaches have also been investigated, with the work of Shah et al. [43] and Vlas and Robinson [44] demonstrating the promise of ontology-driven NLP integration to enhance requirements engineering tasks. Binkhonain and Zhao [45] emphasized that ML algorithm performance is inconsistent across studies and listed three major hurdles in their systematic literature review: a standardized dataset for training, a common definition of NFR, and a defined feature-identifier and feature-selection method used in previous works. Unlike the method proposed in [46], which presents an *ad-hoc* approach to classify NFRs based on experimental datasets and traditional techniques on a limited subset of

PURE datasets, we use the PROMISE dataset, leveraging advanced semantic and syntactic analysis techniques like Word2Vec and BERT, providing a more extensive and flexible classification framework for unrestricted documents.

Overall, existing NFR classification approaches tend to focus on classical ML classifiers, deep learning models, or combinations of the two, but lack a theoretically driven hybridisation strategy. In addition, the design of robust classifiers has received little attention, particularly for small and imbalanced datasets, which are common in requirements engineering practices. In contrast, the novelty of the proposed method lies in combining Word2Vec-based semantic representations with CNN-based deep feature extraction and an Improved SVM-RBF classifier. To improve robustness, interpretability, and generalizability in data-scarce scenarios, the proposed framework does not rely on neural classifiers on top of CNNs, as previous state-of-the-art methods have, but instead uses CNN as a feature abstraction module and a kernel-based classifier to make the final decision. The hybrid design of the proposed NFR classification framework directly addresses the limitations of previous studies and offers a methodologically novel solution for automated NFR classification.

3 Proposed Methodology

This section presents the methodology, approaches, data, and tools used to conduct the research on NFR detection and classification. The methods employed in this investigation are illustrated in Fig. 1.

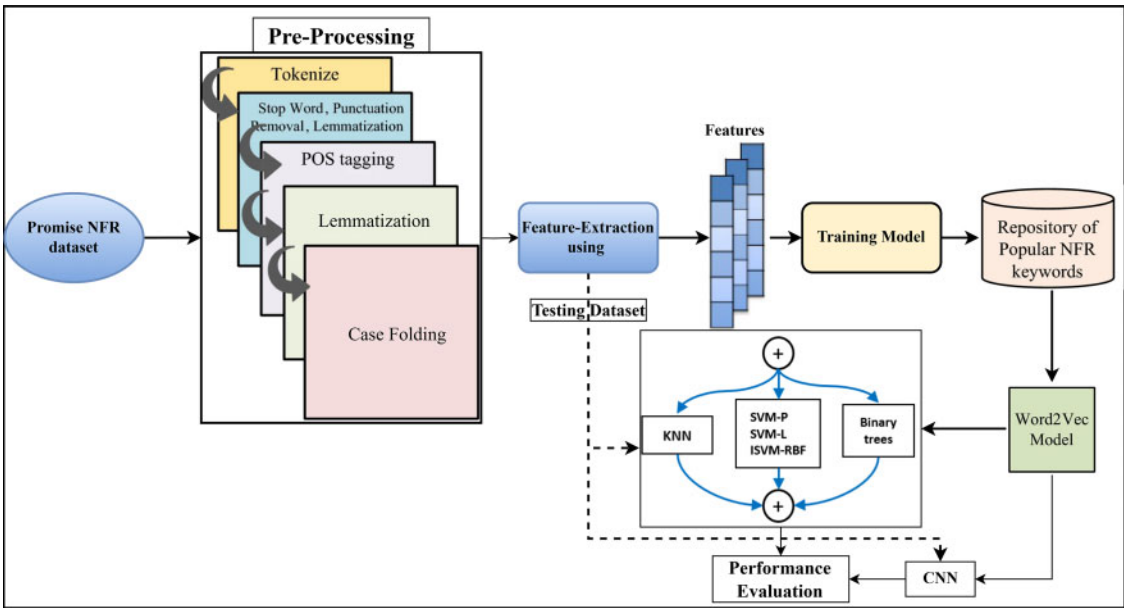

Figure 1: Proposed methodology

3.1 PROMISE NFR Dataset

The proposed approach is investigated on the PROMISE NFR dataset, a publicly available, well-manually annotated benchmark dataset adopted in many requirements engineering research studies. The dataset is publicly accessible at Tera-PROMISE, in the Open Science repository, for transparency and reproducibility. The PROMISE dataset contains 625 natural language requirements:

255 Functional Requirements (FR) and 370 Non-Functional Requirements (NFR). PROMISE is one of the three main datasets that fundamentally differ from requirement datasets such as PURE, as each requirement is annotated as either an FR or one of several categories of NFRs. Requirements are tagged as either FR or NFR, and NFRs are further classified into eleven pre-defined quality attribute categories, as shown in Table 1.

Table 1: PROMISE NFR

NFR description	NFR Label	Abbreviation representation	Count of NFR instances	Balanced NFR count
Usability	US	0	67	67
Performance	PE	1	54	54
Operational	O	2	62	62
Security	SE	3	66	66
Look and Feel	LF	4	38	38
Maintainability	MN	5	17	17
Availability	A	6	21	21
Legal	L	7	13	13
Portability	PO	8	1	1
Scalability	SC	9	21	21
Functional	F	10	255	55
Fault Tolerance	FT	11	10	10

Unlike the PURE dataset, which focuses on recognizing broad requirement classes, the labels in the PROMISE dataset are much more specific. This makes it a standardized benchmark to evaluate NFR-specific classification performance, which is why we have used it here. There is a high degree of class imbalance in the dataset, with some categories, like Portability, having only one instance. To mitigate the negative consequences of this imbalance, a controlled class balancing technique was applied, involving undersampling of majority classes while preserving all minority class instances. This process resulted in a balanced training subset that was consistently applied across all experiments to reduce class imbalance bias. Our proposed framework is model-agnostic and can be extended to other requirement datasets (e.g., PURE); however, we do not provide a direct comparative evaluation against such datasets, as their structures and labeling schemes differ significantly. One of the future research directions suggested in this paper is cross-dataset validation on PROMISE and PURE.

3.2 Pre-Processing and Preparation of Requirement Documents

In the proposed methodology, the first step is the pre-extraction of documents that contain NFRs. This important step involves splitting the complex composition of paragraphs, sentences, words, numbers, punctuation, and special characters into tokens. Pre-processing consists of three broad activities: tokenization, data cleaning, and data normalization, which help prepare the data for analysis.

We carefully segment the documents during data preparation into paragraph- and sentence-level splits. Each sentence begins with a capital letter and ends with a period, question mark, or exclamation

point. It is crucial that this specific segmentation is performed; otherwise, further analyses will lack clarity and confidence.

Requirement Sentence:

The product shall be available during regular business hours. As long as the user has access to the client's PC, the system will be available 99% of the time during the first six months of operation.

Prepared (Tokenized) Sentence:

['The', 'product', 'shall', 'be', 'available', 'during', 'normal', 'business', 'hours', '.', 'As', 'long', 'as', 'the', 'user', 'has', 'access', 'to', 'the', 'client', 'PC", 'the', 'system', 'will','be','available','99', '%o', 'of, 'the', 'time', 'during', 'the', 'first', 'six', 'months', 'of', 'operation,.']

Data cleaning is an important process at this stage, as it improves data quality by eliminating noise components that do not add semantic value to the dataset. This includes punctuation marks, stopwords (frequent words with little semantic value), and non-alphabetic tokens such as numbers and special characters. Using functions from a toolkit like the Natural Language Toolkit (NLTK), we filter out these distractions. The steps involve:

Remove Punctuation: Eliminating punctuation marks of any kind to convert text into meaningful words.

Stop-word and Tag Removal: Removing high frequency but low-value words or tags that have limited semantic significance.

Removal of Non-Alphabetic Tokens: Discarding units of text that are not alphabetic characters, which helps prepare the text for analysis.

Such a holistic treatment ensures that the documents are well-prepared for the extraction and classification of requirements, with an eventual positive impact on feature extraction and model performance.

3.3 Normalization and Vectorization for Feature Extraction

This phase of normalization aims to standardize words to their base forms in order to improve the modeling and matching of texts. The process includes case folding, Part-of-Speech (POS) tagging, and lemmatization, which standardize the text into a consistent format for analysis. The feature extraction phase, after normalization, leverages NLP functions and enables vectorization through the Word2Vec model. In this study, the dot product was used to convert text reports into numerical vectors. We applied Google's Word2Vec method, which represents words as vectors of 300 dimensions. This allows us to convert textual data into a numerical vector representation, which is essential for the classification presented in Fig. 2. In this study, we adapted our method for various classifiers: for traditional ML models like Naive Bayes, Logistic Regression, and Support Vector Machines, we average the sentences into 300-feature vectors. For Convolutional Neural Networks, we pass the sentences in numpy format (2D arrays) and apply padding techniques to keep the array dimensionality consistent (50 words max). Such vectorization creates a structure in the data that makes it easier for ML algorithms to work with, allowing for accurate analysis and classification of requirement sentences.

```
In [6]: sentence = 'the system shall refresh the display every seconds'
        print([model[w] for w in sentence if w in model])

[array([-3.37890625e-01,  1.98242188e-01, -2.96875000e-01,  1.48437500e-01,
        -2.17773438e-01, -3.68652344e-02, -5.82885742e-03, -1.21093750e-01,
         1.42578125e-01, -5.05371094e-02, -5.90820312e-02, -7.42187500e-02,
        -2.51953125e-01, -6.83593750e-02, -1.22558594e-01,  1.63085938e-01,
         2.74658203e-03,  3.61328125e-01, -5.67626953e-03, -1.25976562e-01,
        -2.91015625e-01, -1.12792969e-01,  2.57812500e-01, -7.17773438e-02,
         2.36816406e-02, -7.51953125e-02, -3.67187500e-01,  2.78320312e-02,
        -2.44140625e-01, -8.30078125e-03,  1.23046875e-01,  1.20605469e-01,
         9.33837891e-03,  8.85009766e-03, -1.00585938e-01,  1.98242188e-01,
        -4.92187500e-01,  1.47460938e-01,  1.52343750e-01,  1.25000000e-01,
        -8.30078125e-02, -1.52343750e-01,  1.75781250e-01, -5.54199219e-02,
         4.39453125e-03, -3.36914062e-02,  7.08007812e-02, -5.37109375e-02,
        -1.24511719e-01,  2.34375000e-01, -1.40625000e-01,  2.36328125e-01,
        -1.53320312e-01,  6.13281250e-01,  2.94921875e-01, -9.52148438e-02,
         6.49414062e-02, -1.57226562e-01, -1.62109375e-01, -1.25000000e-01,
        -1.55273438e-01,  5.51757812e-02, -4.51660156e-02, -9.42382812e-02,
        -4.79125977e-03, -2.75390625e-01, -2.30468750e-01,  1.47705078e-02,
         1.68457031e-02,  1.10839844e-01,  5.07812500e-02, -7.17773438e-02,
        -2.67333984e-02, -8.64257812e-02, -1.86523438e-01, -5.05371094e-02,
```

Figure 2: Requirement sentence representation in the W2V model

3.4 Enhanced Machine Learning Framework for NFR Detection

The proposed system comprises a set of ML models that convert requirement sentences into numerical vectors, which are then classified into their respective Non-Functional Requirement (NFR) categories, including usability, availability, reliability, security, and performance. We implement a mixture of high-end ML algorithms, including different types of Support Vector Machine (SVM) kernels, such as RBF, Linear, and Polynomial kernels, as well as K-Nearest Neighbors (KNN) and Decision Trees (DT). A voting mechanism enhances the consensus of these models by demonstrating the classification results based on the majority rule for each NFR category.

To learn reliably with a large class imbalance and limited data, all traditional ML models were trained and evaluated using stratified k-fold cross-validation, ensuring that class distributions in each fold remain the same as in the full dataset.

We then enhance traditional classifiers with a Convolutional Neural Network (CNN), which has proved to be an effective feature extractor. Although CNNs are primarily used for image data, our data is in text form. The CNN converts the sentences into vectors, and through one or more convolution layers, it extracts key features that improve performance and deep classification. The PROMISE dataset is small, which motivated the choice of a specific architecture and regularization techniques to prevent CNN overfitting. These techniques include limiting the CNN to a shallow topology, using dropouts, applying L2 regularization, and implementing early stopping when the validation loss converges.

The ISVM-RBF (Improved Support Vector Machine with Radial Basis Function) is one of our working real-time classifiers in the new suite. This sophisticated algorithm is specifically designed to handle the large datasets typical of NFRs. Unlike deep learning models, ISVM-RBF performs

exceptionally well for small, high-dimensional textual datasets and is a promising model for data-scarce scenarios. The ISVM-RBF uses a kernel function that projects data points into a higher-dimensional space, where classification becomes easier, especially when the data is not linearly separable. The ISVM-RBF can be represented as follows:

$$ISVM - RBF = \forall\omega\,(a, a_i) = \exp\left(-\frac{1}{\sigma^2}\,||a, a_i||^2\right) \times \lambda \tag{1}$$

where $\forall\omega\,(a, a_i)$ represents the interaction between data points in their transformed space, adjusting the fit to improve detection efficiency and precision. ISVM-RBF exhibits strong non-linear capacity, and since the kernel functions are symmetric, it has the ability to accurately extract spatial features that are critical for practical applications. Moreover, to build a confidence report for multi-class states, ISVM-RBF adopts a one-vs-all scheme for the binary case. In this scheme, the class that generates the maximum probability is used to determine the final state from the individual one-vs-all classifiers. Thus, powerful ML classifiers are combined in this approach to build a robust detection system for NFR classification. The framework demonstrates stable and consistent performance across the PROMISE dataset. However, generalization is still limited to the same dataset due to the lack of other publicly available NFR datasets with the same labeling scheme. Our system integrates the depth feature understanding capabilities of CNNs with the interpretability and verified efficacy of conventional ML methods, combining traditional ML techniques with CNNs. It is effective with the sparsity of NLP data and high-dimensional spaces, directly improving classification accuracy and effectiveness. The algorithmic workflow for NFR text classification using a pretrained Word2Vec model followed by a CNN is summarized in Table 2. This workflow outlines the steps in preprocessing, feature extraction, and classification, summarizing the methodology of this study.

Table 2: Algorithmic workflow for NFR text classification using pretrained Word2Vec and CNN

Stage	Algorithm description (Formal Pseudocode Representation)	Outcome
Input Definition	Load the PROMISE NFR dataset containing requirement text and twelve binary NFR class indicators {US, PE, O, SE, LF, MN, A, L, PO, SC, F, FT}.	Raw dataset D
Data Acquisition	Read the dataset from CSV format using a semicolon delimiter and store it in a structured dataframe.	Dataframe D
Text Normalization	For each requirement description: convert text to lowercase; remove punctuation, numerical values, HTML tags, and non-alphabetic symbols using regular expressions.	Cleaned text corpus
Feature Augmentation	Append the normalized text as a new attribute (clean) to the original dataset for downstream processing.	Enhanced dataframe
Embedding Model Loading	Load the pretrained Google News Word2Vec model with 300-dimensional embeddings.	Embedding model W

(Continued)

Table 2 (continued)

Stage	Algorithm description (Formal Pseudocode Representation)	Outcome
Sentence Vector Construction	For each cleaned sentence, initialize a zero vector of dimension 300 and compute the sentence embedding by summing the embeddings of constituent tokens. Tokens absent from the vocabulary are ignored.	Vectorized text representations
Label Encoding	For each instance, identify the active NFR class by locating the index of value 1 across the twelve binary label columns.	Integer-encoded labels (0–11)
Class Distribution Analysis	Compute and report class frequency statistics to identify imbalance across NFR categories.	Label distribution summary
Class Balancing	Reduce dataset bias by removing a predefined number of samples from the dominant class (class index 10) using deterministic down-sampling.	Balanced dataset
Dataset Partitioning	Randomly divide the dataset into training and testing subsets using a 70:30 split ratio.	Training and testing sets
Label Transformation	Convert integer class labels into one-hot encoded vectors for multi-class classification.	One-hot encoded labels
Input Reshaping	Reshape sentence embeddings into a four-dimensional tensor of size $(82 \times 300 \times 1)$ to meet CNN input requirements.	CNN-compatible tensors
Model Architecture Design	Construct a deep CNN comprising three convolutional blocks (Conv2D + Batch Normalization + Max Pooling), followed by feature concatenation, flattening, fully connected layers, and a softmax classifier.	CNN model
Model Compilation	Configure the model using the Adam optimizer, categorical cross-entropy loss function, and classification accuracy as the evaluation metric.	Compiled model
Training Strategy	Train the model for 50 epochs using mini-batch gradient descent with learning-rate decay and validation-based checkpointing.	Optimized model
Output	Store the best-performing CNN weights and record the training history for performance analysis.	Final trained model

3.5 Performance Metrics

Classifier performance in ML is assessed using several key metrics that reflect different aspects of accuracy and prediction reliability. Given the limited size and class imbalance of the PROMISE

dataset, multiple complementary metrics were employed to provide a balanced and reliable evaluation, rather than relying solely on overall accuracy. The metrics include:

- True Positives (TP): Correct predictions of positively labeled samples.
- False Positives (FP): Incorrect predictions where negative samples are incorrectly labeled as positive.
- True Negatives (TN): Correct predictions of negatively labeled samples.
- False Negatives (FN): Incorrect predictions where positive samples are incorrectly labeled as negative.

To ensure robustness under data scarcity, all performance metrics were computed as average values across stratified k-fold cross-validation, which preserves class proportions in each fold and reduces estimation bias.

These metrics are used to calculate the following key performance indicators, which are presented in Table 3:

Table 3: Performance evaluation matrices.

Metric	Description	Formula
Accuracy	Indicates the overall rate of correct predictions across all categories.	$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$
F1-Score	The harmonic mean of precision and recall, balancing both.	$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \times 100\%$
Precision	The proportion of positive predictions that are correctly identified.	$Precision = \frac{TP}{TP + FP} \times 100\%$
Recall	The proportion of actual positives that are correctly identified.	$Recall = \frac{TP}{TP + FN} \times 100\%$

Although these metrics effectively capture classifier performance within the PROMISE dataset, the reported results reflect within-dataset generalization only. Cross-dataset performance evaluation is not included due to the lack of publicly available NFR datasets with compatible class definitions, which is identified as future work.

4 Experimental Results

This paper explores the use of Natural Language Processing (NLP) and Machine Learning (ML) methods for Non-Functional Requirements (NFRs) classification from free-form text. Using the word2vec feature extraction technique, we analyzed various ML models based on the PROMISE dataset. The models examined include Improved Support Vector Machines with Radial Basis Function (ISVM-RBF), Polynomial and Linear SVMs, Decision Tree (DT), K-Nearest Neighbor (KNN), and Convolutional Neural Network (CNN). We used Word2Vec to convert textual requirement sentences into corresponding numerical vectors for our classifiers, which facilitated better training and evaluation results. We measured the performance of the models using accuracy, precision, recall, and F1-score, and classified the final output using a voting-based approach based on the majority of votes from each model. Results showed that NLP and ML approaches could be promising techniques

for NFR identification, with Word2Vec substantially improving the model performance due to sophisticated data preprocessing.

GPU-based experiments were run in the Jupyter Notebook environment to speed up computations. The system configuration included an Intel(R) Core(TM) i9-4210H CPU @ 2.90 GHz, 32 GB RAM, 1024 GB HDD. The system had 39,424 KB Level 3 Cache, was running Windows 10 Pro, and offered ample power to handle more computationally intensive tasks. The availability of Python libraries within the Jupyter Notebook for effective implementation of NLP methods and conducting experiments greatly facilitated this process.

The detailed hyperparameter configurations of all base classifiers, including CNN, SVM, KNN, and Decision Tree models, along with their selection strategy, are summarized in Table 4 to ensure reproducibility and transparency.

Table 4: Hyperparameter settings for base classifiers (CNN, SVM, KNN, DT) and tuning protocol

Hyperparameter	Final value used
	CNN
Input shape	(82, 300, 1)
Conv filters (all 3 layers)	200
Kernel size (all 3 layers)	(2, 10)
Activation	ReLU
L2 regularization	0.01
Batch normalization	Enabled
MaxPool size (all 3 layers)	(2, 2)
Dropout	0.1 (twice)
Dense units	512
Output activation	Softmax
Optimizer	Adam
Learning rate	1×10^{-4}
β1/β2/ε	$0.9/0.999/1 \times 10^{-8}$
Loss	Categorical cross-entropy
Batch size/Epochs	32/50
LR scheduler	(lr\leftarrow lr/(1 + 0.01\cdot epoch))
	SVM (RBF)
Regularization (C)	Grid/validation over: {0.1, 1, 10, 100}
Kernel width (γ)	Grid/validation over: {1×10^{-3}, 1×10^{-2}, 1×10^{-1}, 1} or {“scale”, “auto”}
Class weighting	Use “balanced” if imbalance remains after downsampling

(Continued)

Table 4 (continued)

Hyperparameter	Final value used
	KNN
Number of neighbors (k)	Validation over odd k: {3, 5, 7, 9, 11, 15}
Distance metric	Common: cosine or Euclidean (state explicitly)
Weighting	"uniform" vs. "distance" (choose by validation)
	Decision Tree (DT)
Split criterion	Gini vs. Entropy (choose by validation)
Max depth	Validation over: {None, 10, 20, 30, 40}
Min samples split	Validation over: {2, 5, 10}
Min samples leaf	Validation over: {1, 2, 5}
	Common Protocol
Train:test split	70:30, seed = 42
Feature representation	Pretrained Word2Vec (300-d), sentence vector = sum
Class balancing	Remove 200 samples from class index 10
Selection rule	Best validation macro-F1/val accuracy

4.1 Optimal ML Classifier Using W2V Model

For the experiment, we used the Google Word2Vec model to convert words in requirement sentences into vectors. We employed three classical classifiers—SVM, Decision Tree, and K-Nearest Neighbors- to train them on the vectors of each sentence, which were computed by averaging the word vectors of each sentence. The training dataset vectors were used to train our classifiers, and the testing dataset vectors were used to evaluate them. We structured our word vectors in two-dimensional arrays, similar to how image data is handled, to leverage the inputs for training and testing our Convolutional Neural Network (CNN) classifier. Each experiment was performed ten times, and the mean and median of the classification results for all classifiers are presented in Fig. 3.

Additionally, we calculated the average precision, recall, and F1 score for each type of Non-Functional Requirement (NFR) based on their performance using the Word2Vec model. Class-level performance metrics are summarized in Fig. 4. These values analyze the effectiveness of our model's classification and recognition for each NFR type, providing insights into the precision and strength of our classification approach.

Figure 3: ML and CNN accuracy using the W2V model.

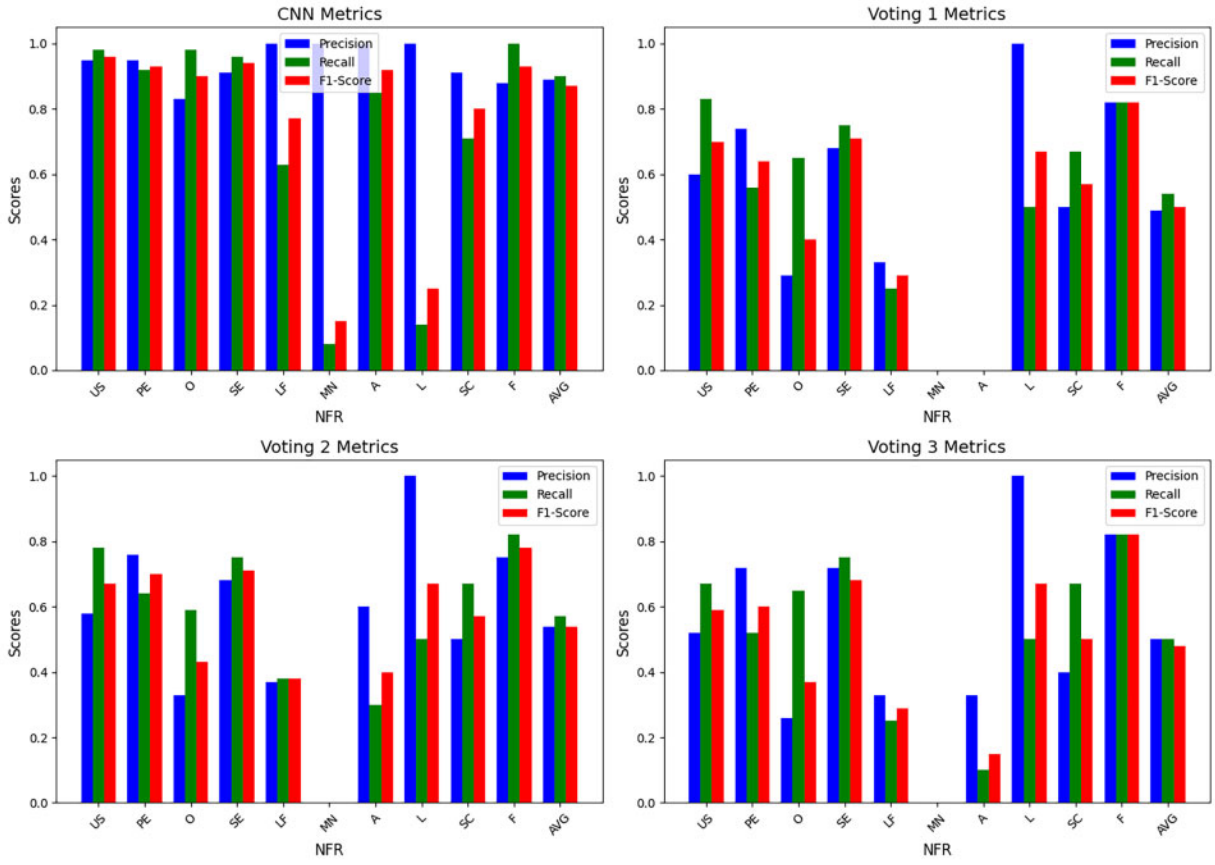

Figure 4: Ml Performance metrics using the W2V model.

4.2 NFR Classification Accuracy Using Different ML Techniques

Following the four experiments, we summarized the average accuracies of all the classifiers based on different NLP feature extraction methods. For each classifier, the balance between correct and incorrect predictions within the test dataset is visualized. In addition, Fig. 5 provides a summary of

these results, showing a comparison of the mean accuracy for each classifier. This visualization helps to understand how different NLP techniques used in the experiments contribute to improving the performance of the classifiers.

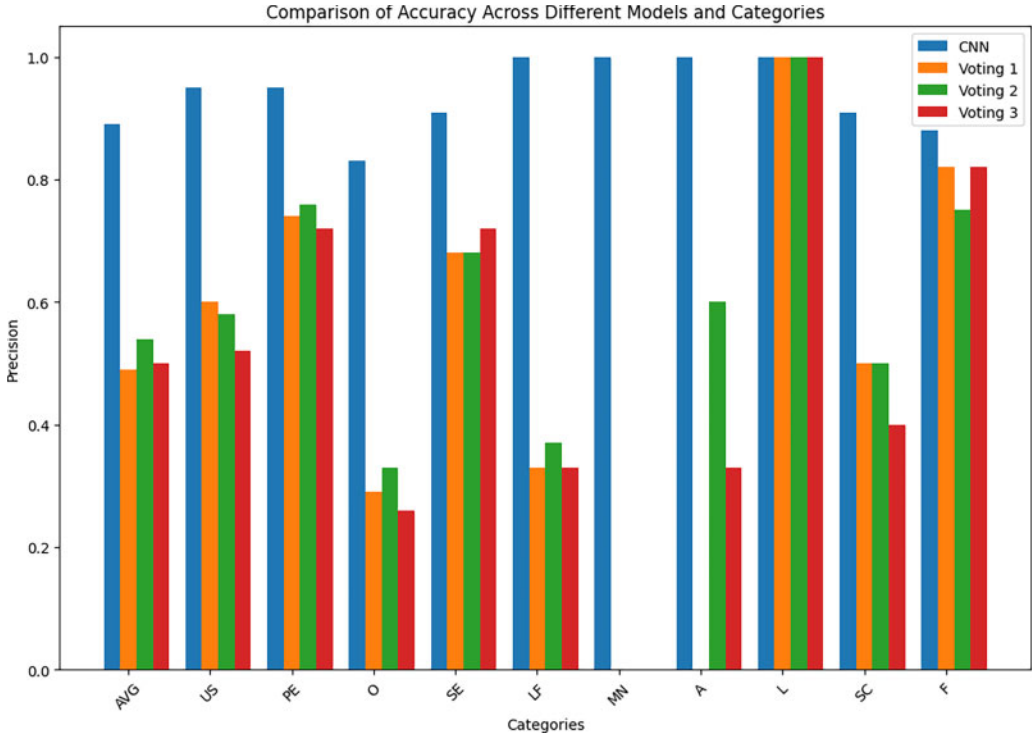

Figure 5: ML classifiers accuracy using the NLP method

The CNN method has been found to produce the best accuracy compared to other tested NLP-based feature extraction methods in our exploration of NFR classification. During the initial CNN training, apart from using random epochs, we studied how many iterations it would take to reach an acceptable classification accuracy. After several fine-tunings of the training configuration, the classification accuracy reached 89%, indicating excellent performance. Specifically, the training and validation accuracy rates are depicted in Fig. 6a and 6b. The trends of accuracy and validation loss during the training process are also shown in Fig. 6, providing a complete view of the model's performance over time.

Fig. 7 compares the multi-class classification model using the ROC curve. The plot contains ROC curves for all classes, showing the True Positive Rate (TPR) and False Positive Rate (FPR) at different thresholds. These curves are plotted individually for each class and indicate the classifier's performance as a function of the Area Under the Curve (AUC). Random guessing is shown as a baseline by the inclusion of a diagonal line. The ROC AUC is a vital metric used to assess the discriminative power of the model, which is calculated and reported for each class. The confusion matrix is an effective tool for visually analyzing the classifier's performance in identifying more than two classes.

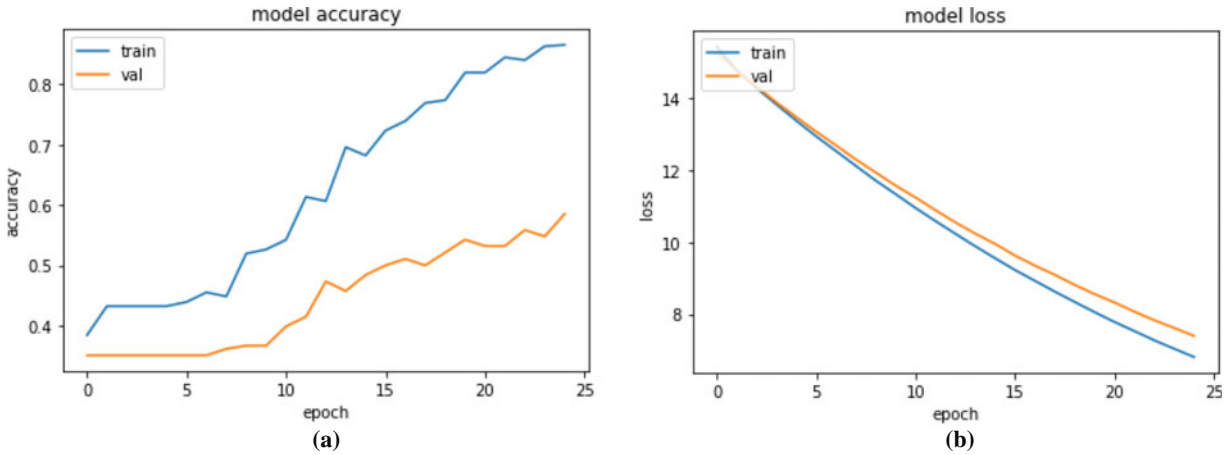

Figure 6: (**a**) Accuracy. (**b**) Loss

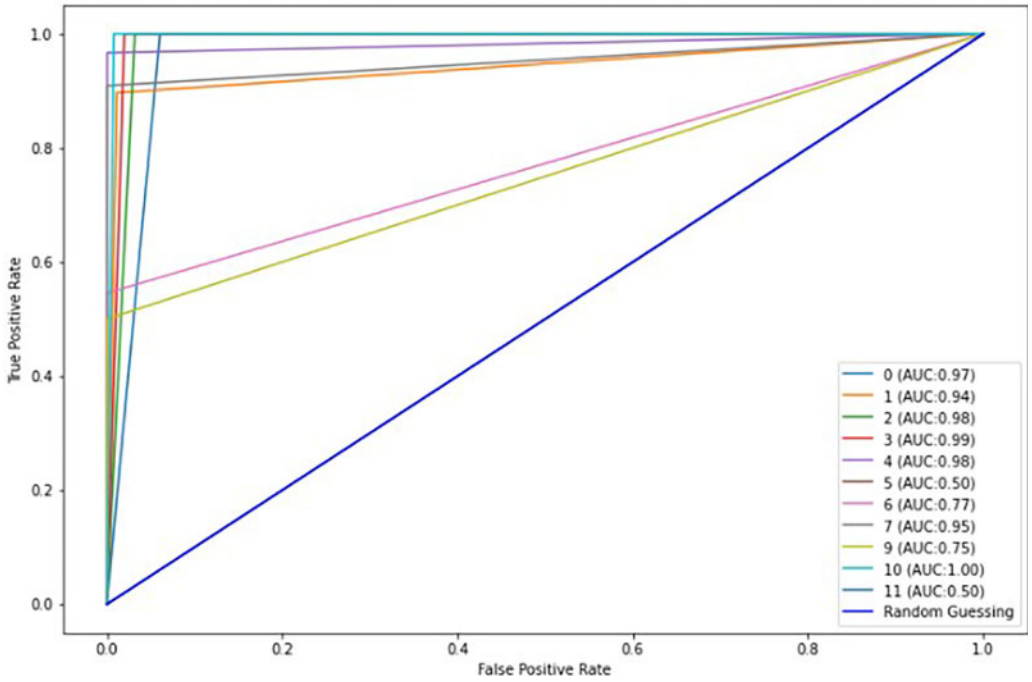

Figure 7: ROC curves for the multi-class classification model

5 Discussion

This paper assesses several ML methods to classify Non-Functional Requirement (NFRs). We compared classical classifiers, such as SVM (with various kernels), K-Nearest Neighbors, and Decision Trees, with a Convolutional Neural Network (CNN) using the Word2Vec method. Preliminary results from the initial classifier were relatively encouraging, and as we moved from one classifier to two, and then to three classifiers, we saw continued improvements across several metrics, as illustrated in Fig. 8. The CNN performed the best, particularly as it was able to handle multi-dimensional input vectors and extract features effectively.

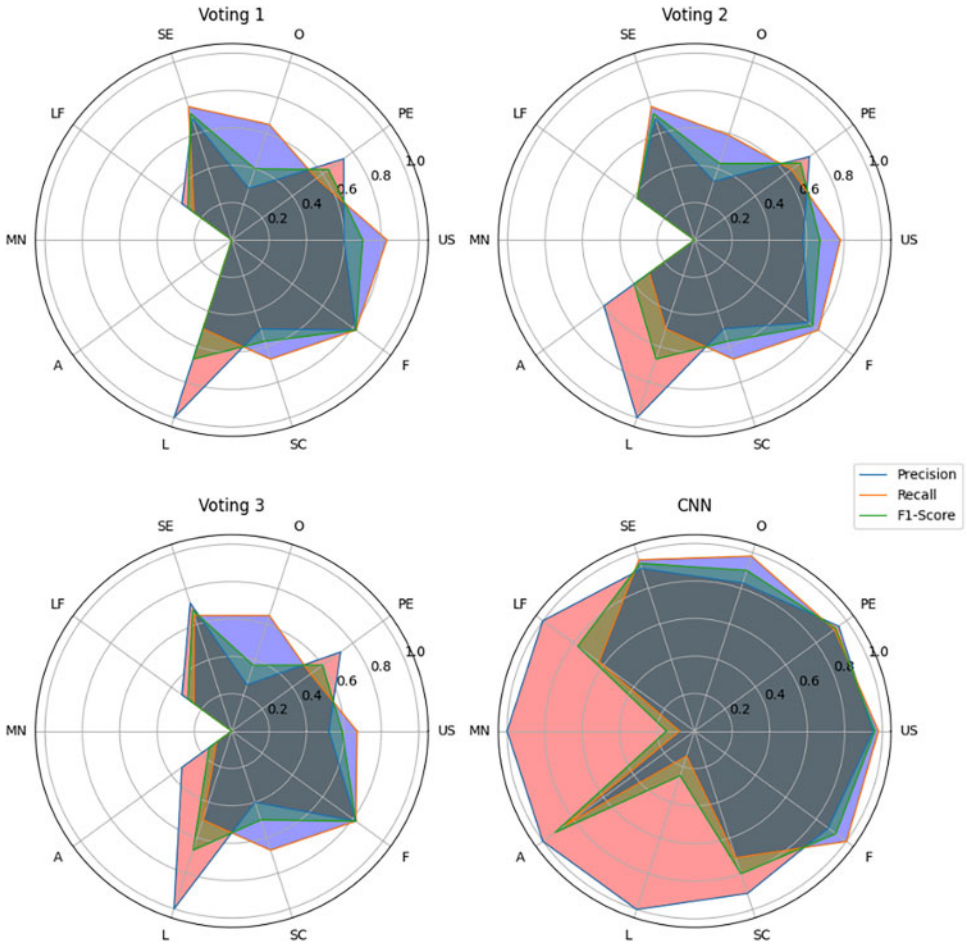

Figure 8: Voting 1,2,3 and CNN results

The performance of NFR classification is highly influenced by both the classifier employed and the NLP feature extraction techniques used. Although the trends for precision, recall, and F1-Score were mostly consistent with the accuracy trend across most NFR categories, there were exceptions, particularly in the case of the Maintainability category. In this case, precision outperformed traditional classifiers, with the CNN achieving a precision of 100%. Traditional classifiers achieved up to 60% precision per class, while the CNN reached up to 90%. On the upside, the CNN also achieved an impressive recall rate of 91.4%, meaning that out of every 100 relevant NFRs, the CNN correctly identified 91 of them.

Additionally, we outperform previous works on the PROMISE NFR dataset, as shown in Table 5. As a result, we achieved 89% precision, 90% recall, and an F1-score of 87%, which represents a significant improvement over the other studies we referred to. This enhancement suggests that our approach is effective not only on the PROMISE dataset but also applicable to other datasets, such as the CCHIT dataset, where requirement sentences may contain multiple NFR types. Interestingly, our method did not produce separate results on the CCHIT dataset but only reported improvements on the PROMISE NFR dataset.

Table 5: Comparative analysis with an existing study on PROMISE NFR dataset

Ref #	Precision %	Recall%	F1-Score%
[3]	72	72	70
[12]	78	81	77
[16]	77	67	68
[47]	86	88	87
[41]	71	71	70
Proposed	89	90	87

6 Conclusions

In this paper, we presented an automated system based on cutting-edge ML and NLP approaches that generates classifications for ten types of NFRs from unstructured requirement documents. Using traditional and novel word embedding techniques (Word2Vec), our method employed multiple ML classifiers with enhanced SVMs and CNN models, with performance tested on the PROMISE dataset.

The results of this study show that while traditional ML models performed moderately well, CNNs performed significantly better, achieving a precision score of 90%. In summary, CNNs outperformed other classifiers due to the nature of feature extraction with CNNs, and word embedding also proved to represent the software requirements with better alignment to the NFR classification compared to classical NLP-based methods. These findings will be part of our future work, where we plan to improve NFR classification by increasing the number of NFR classes and testing other ML approaches, such as Recurrent Neural Networks (RNNs) and model fusion. Finally, we will test our methods on more datasets to enhance the classification's accuracy and the system's robustness against new data samples.

Acknowledgement: Not applicable.

Funding Statement: This work was supported and funded by the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University (IMSIU) (grant number IMSIU-DDRSP2604).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Muhammad Shafiq and Waeal J. Obidallah; methodology, Muhammad Shafiq; software, Muhammad Shafiq; validation, Waeal J. Obidallah, Mubarak Albathan and Tahir Kamal; formal analysis, Muhammad Shafiq; investigation, Muhammad Shafiq; resources, Waeal J. Obidallah and Mubarak Albathan; data curation, Tahir Kamal; writing—original draft preparation, Muhammad Shafiq; writing—review and editing, Waeal J. Obidallah and Tahir Kamal; visualization, Muhammad Shafiq; supervision, Waeal J. Obidallah; project administration, Waeal J. Obidallah; funding acquisition, Waeal J. Obidallah. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly available at https://github.com/tobhey/NoRBERT.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Sommerville I. Software engineering. 9th ed. Upper Saddle River, NJ, USA: Pearson; 2011.
2. Mead NR, Stehney T. Security quality requirements engineering (SQUARE) methodology. ACM SIGSOFT Softw Eng Notes. 2005;30(4):1–7. doi:10.1145/1082983.1083214.
3. Rashwan A, Ormandjieva O, Witte R. Ontology-based classification of non-functional requirements in software specifications: a new corpus and SVM-based classifier. In: Proceedings of the 2013 IEEE 37th Annual Computer Software and Applications Conference; 2013 Jul 22–26. Kyoto, Japan. doi:10.1109/compsac.2013.64.
4. van Lamsweerde A. Goal-oriented requirements engineering: from system objectives to UML models to precise software specifications. In: Proceedings of the 25th International Conference on Software Engineering, 2003. Proceedings; 2003 May 10; Portland, OR, USA. doi:10.1109/icse.2003.1201266.
5. Borg A, Yong A, Carlshamre P, Sandahl K. The bad conscience of requirements engineering: an investigation in real-world treatment of non-functional requirements. In: Proceedings of the 3rd Conference on Software Engineering Research and Practice in Sweden (SERPS); 2003 Oct 23–24; Lund, Sweden.
6. Ameller D, Ayala C, Cabot J, Franch X. How do software architects consider non-functional requirements: an exploratory study. In: Proceedings of the 2012 20th IEEE International Requirements Engineering Conference (RE); 2012 Sep 24–28; Chicago, IL, USA. doi:10.1109/re.2012.6345838.
7. Sommerville I. YAZILIM MÜHENDİSLİĞİ-software engineering. Ankara, Turkey: Nobel Akademik Yayıncılık; 2016.
8. Wang T, Hou B, Li J, Shi P, Zhang B, Snoussi H. TASTA: text-assisted spatial and temporal attention network for video question answering. Adv Intell Syst. 2023;5(4):2200131. doi:10.1002/aisy.202200131.
9. Huang F, Bei Y, Yang Z, Jiang J, Chen H, Shen Q, et al. Large language model simulator for cold-start recommendation. In: Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining. Hannover Germany: ACM; 2025. doi:10.1145/3701551.3703546.
10. Zhang Z-W, Liu Z-G, Martin A, Zhou K. BSC: belief shift clustering. IEEE Trans Syst Man Cybern Syst. 2022;53(3):1748–60.
11. Zhao HH, Ji TL, Rosin PL, Lai YK, Meng WL, Wang YN. Cross-lingual font style transfer with full-domain convolutional attention. Pattern Recognit. 2024;155(2):110709. doi:10.1016/j.patcog.2024.110709.
12. Kurtanovic Z, Maalej W. Automatically classifying functional and non-functional requirements using supervised machine learning. In: Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE); 2017 Sep 4–8; Lisbon, Portugal. doi:10.1109/re.2017.82.
13. Lu Y, Yang J. Quantum financing system: a survey on quantum algorithms, potential scenarios and open research issues. J Ind Inf Integr. 2024;41(2):100663. doi:10.1016/j.jii.2024.100663.
14. Slankas J, Williams L. Automated extraction of non-functional requirements in available documentation. In: Proceedings of the 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE); 2013 May 25; San Francisco, CA, USA. doi:10.1109/naturalise.2013.6611715.
15. Huang Y, Wu J, Feng Y, Chen Z, Zhao Z. An empirical study on clustering for isolating bugs in fault localization. In: Proceedings of the 2013 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW); 2013 Nov 4–7; Pasadena, CA, USA. doi:10.1109/issrew.2013.6688893.
16. Amasaki S, Leelaprute P. The effects of vectorization methods on non-functional requirements classification. In: Proceedings of the 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA); 2018 Aug 29–31; Prague, Czech Republic. doi:10.1109/seaa.2018.00036.
17. Tóth L, Vidács L. Study of various classifiers for identification and classification of non-functional requirements. In: International Conference on Computational Science and Its Applications. Cham, Switzerland: Springer International Publishing; 2018. p. 492–503. doi:10.1007/978-3-319-95174-4_39.
18. Li S, Yang J, Bao H, Xia D, Zhang Q, Wang G. Cost-sensitive neighborhood granularity selection for hierarchical classification. IEEE Trans Knowl Data Eng. 2025;37(8):4471–84. doi:10.1109/tkde.2025.3566038.

19. Yin L, Wang L, Cai Z, Lu S, Wang R, AlSanad A, et al. DPAL-BERT: a faster and lighter question answering model. Comput Model Eng Sci. 2024;141(1):771–86. doi:10.32604/cmes.2024.052622.
20. Zheng W, Lu S, Cai Z, Wang R, Wang L, Yin L. PAL-BERT: an improved question answering model. Comput Model Eng Sci. 2024;139(3):2729–45. doi:10.32604/cmes.2023.046692.
21. Cao Z, Huang L, Wang T, Wang Y, Shi J, Zhu A, et al. Understanding the dimensional need of noncontrastive learning. IEEE Trans Cybern. 2025;55(9):4089–102. doi:10.1109/tcyb.2025.3577745.
22. Meng Q, Song Y, Mu J, Lv Y, Yang J, Xu L, et al. Electric power audit text classification with multi-grained pre-trained language model. IEEE Access. 2023;11(11):13510–8. doi:10.1109/access.2023.3240162.
23. Huang M, Shen A, Li K, Peng H, Li B, Su Y, et al. EdgeLLM: a highly efficient CPU-FPGA heterogeneous edge accelerator for large language models. IEEE Trans Circuits Syst I. 2025;72(7):3352–65. doi:10.1109/tcsi.2025.3546256.
24. Shen X, Li L, Ma Y, Xu S, Liu J, Yang Z, et al. VLCIM: a vision-language cyclic interaction model for industrial defect detection. IEEE Trans Instrum Meas. 2025;74:1–13. doi:10.1109/tim.2025.3583364.
25. Wu L, Wang K, Nie K, Guo S, Gao C, Wang Z, et al. TFGIN: tight-fitting graph inference network for table-based fact verification. ACM Trans Inf Syst. 2025;43(5):1–26. doi:10.1145/3734520.
26. Eli E, Wang D, Xu W, Mamat H, Aysa A, Ubul K. A comprehensive review of non-Latin natural scene text detection and recognition techniques. Eng Appl Artif Intell. 2025;156(6):111107. doi:10.1016/j.engappai.2025.111107.
27. Wu X, He Z, Jiang G, Yu M, Song Y, Luo T. No-reference point cloud quality assessment through structure sampling and clustering based on graph. IEEE Trans Broadcast. 2025;71(1):307–22. doi:10.1109/tbc.2024.3482173.
28. Liu W, Chen X, Miao D, Zhang H, Qin X, Du S, et al. SEAD-MGFE-Net: schrödinger equation-based adaptive dropout multi-granular feature enhancement network for conversational aspect-based sentiment quadruple analysis. Inf Sci. 2026;723(11):122684. doi:10.1016/j.ins.2025.122684.
29. Cao X, Xu M, Yu X, Yao J, Ye W, Huang S, et al. Analytical survey of learning with low-resource data: from analysis to investigation. ACM Comput Surv. 2026;58(6):1–47. doi:10.1145/3773075.
30. Yin Z, Wang S. Span-level detection of AI-generated scientific text via contrastive learning and structural calibration. Knowl Based Syst. 2026;334(7945):115123. doi:10.1016/j.knosys.2025.115123.
31. Huang D, Shi X, Feng Y, Zhao C, Wen J, Shang M. Automatic compiler tuning for inlining with machine learning. Int J Patt Recogn Artif Intell. 2025;39(11):2552016. doi:10.1142/s0218001425520160.
32. Liu M, Ma W, Wang C, Wang P, Yang M. Machine learning-enhanced rapid design of hydrodynamic shape for underwater vehicles pedigrees. IEEE/ASME Trans Mechatron. 2025:1–11. doi:10.1109/tmech.2025.3639691.
33. Wang T, Wang Z, Li H, Xia C, Zhao C. HHG-bot: a hyperheterogeneous graph-based twitter bot detection model. IEEE Trans Comput Soc Syst. 2025;12(5):3416–30. doi:10.1109/tcss.2025.3543419.
34. Wang T, Liu M, Li H, Zhao L, Jiang C, Xia C, et al. ArchSentry: enhanced Android malware detection via hierarchical semantic extraction. IEEE Trans Netw Serv Manage. 2025;22(3):2822–37. doi:10.1109/tnsm.2025.3559255.
35. Wang Z, Gao Z, Yang Y, Wang G, Jiao C, Shen HT. Geometric matching for cross-modal retrieval. IEEE Trans Neural Netw Learning Syst. 2025;36(3):5509–21. doi:10.1109/tnnls.2024.3381347.
36. Jiang JY, Tsai SC, Lee SJ. FSKNN: multi-label text categorization based on fuzzy similarity and k nearest neighbors. Expert Syst Appl. 2012;39(3):2813–21. doi:10.1016/j.eswa.2011.08.141.
37. Ramadhani DA, Rochimah S, Yuhana UL. Classification of non-functional requirements using semantic-FSKNN based ISO/IEC 9126. TELKOMNIKA Telecommun Comput Electron Control. 2015;13(4):1456. doi:10.12928/telkomnika.v13i4.2300.

38. Winkler J, Vogelsang A. Automatic classification of requirements based on convolutional neural networks. In: Proceedings of the 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW); 2016 Sep 12–16; Beijing, China. doi:10.1109/rew.2016.021.
39. Baker C, Deng L, Chakraborty S, Dehlinger J. Automatic multi-class non-functional software requirements classification using neural networks. In: Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC); 2019 Jul 15–19; Milwaukee, WI, USA. doi:10.1109/compsac.2019.10275.
40. Dekhtyar A, Fong V. RE data challenge: requirements identification with Word2Vec and TensorFlow. In: Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE); 2017 Sep 4–8; Lisbon, Portugal. doi:10.1109/re.2017.26.
41. Rahman MA, Haque MA, Tawhid MNA, Siddik MS. Classifying non-functional requirements using RNN variants for quality software development. In: Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation; 2019 Aug 27; Tallinn, Estonia. doi:10.1145/3340482.3342745.
42. Casamayor A, Godoy D, Campo M. Identification of non-functional requirements in textual specifications: a semi-supervised learning approach. Inf Softw Technol. 2010;52(4):436–45. doi:10.1016/j.infsof.2009.10.010.
43. Shah US, Patel SJ, Jinwala D. Specification of non-functional requirements: a hybrid approach. In: Proceedings of the REFSQ Workshops; 2016 Mar 14–17; Gothenburg, Sweden.
44. Vlas R, Robinson WN. A rule-based natural language technique for requirements discovery and classification in open-source software development projects. In: Proceedings of the 2011 44th Hawaii International Conference on System Sciences. 2011 Jan 4–7; Kauai, HI, USA. doi:10.1109/hicss.2011.28.
45. Binkhonain M, Zhao L. A review of machine learning algorithms for identification and classification of non-functional requirements. Expert Syst Appl X. 2019;1(2):100001. doi:10.1016/j.eswax.2019.100001.
46. Shreda QA, Hanani AA. Identifying non-functional requirements from unconstrained documents using natural language processing and machine learning approaches. IEEE Access. 2025;13(1):124159–79. doi:10.1109/access.2021.3052921.
47. Hey T, Keim J, Koziolek A, Tichy WF. NoRBERT: transfer learning for requirements classification. In: Proceedings of the 2020 IEEE 28th International Requirements Engineering Conference (RE); 2020 Aug 31–Sep 4; Zurich, Switzerland. doi:10.1109/re48521.2020.00028.