Requirement Classification using Deep Learning and Nature-Inspired Optimization Technique

Sonal Sonawane, Shubha Puthran
Department of Computer Engineering, Mukesh Patel School of Technology Management & Engineering, SVKM's NMIMS, Mumbai-400056, India,
E-mail: sonalraut.shelly@gmail.com1, shubha.puthran@nmims.edu2

Keywords: requirement engineering, software requirement, natural language processing, convolutional neural network, flower pollination optimizer, machine learning

Received: September 4, 2024

Recent advancements in requirements engineering (RE) methods have increasingly leveraged machine learning (ML) algorithms to address challenging RE issues such as the identification and classification of software requirements in requirement documents. Classifying software requirements accurately and efficiently is crucial for the success of the project. Functional and nonfunctional requirements define the core attributes and constraints of a system. When requirements are documented in natural language, they often introduce uncertainties such as ambiguity, inconsistency, or poor readability making the process challenging. Further, manual extraction of requirements is tideous and error prone as it requires precise interpretation, thereby mitigating the risks of miscommunication and errors in the development process.To address these challenges, it is vital to employ Natural Language Processing (NLP)techniques to enhance the clarity of requirements. This paper proposes a method that combines deep learning models with NLP techniques, supplemented by Flower Pollination Optimizer (FPO) algorithm, to automate the classification of requirements. The methodology leverages NLP to extract significant features for training a Convolutional Neural Network (CNN) model. CNN model is enhanced using the FPO algorithm to ensure better convergence. The performance of the proposed CNN-FPO framework is evaluated using an industry SmartNet dataset, with results indicating that the accuracy of the classification can reach up to 98% as compared to other traditional machine learning(ML) approaches.The integration of NLP with an optimized CNN model creates a robust framework for requirement classification, addressing the challenges posed by natural language documentation in requirements engineering.

Povzetek: Predlagana je metoda za samodejno klasifikacijo programskih zahtev s kombinacijo konvolucijske nevronske mreže (CNN), obogatene z algoritmom Flower Pollination Optimizer (FPO), in obdelavo naravnega jezika (NLP).

1 Introduction

Requirements Engineering (RE) can be defined as "a set of activities for exploring, evaluating, documenting, revising and adapting the objective, constraints and assumptions that the system should meet" [1]. These activities are recorded in a document known as the requirement document (RD), which is intended for communication, analysis, and experimentation. These requirement document are written in natural language and can be processed as any text document. It contains paragraphs, sentences, and words.There are two types of requirement document unconstrained document and disciplined or constrained document. Unconstrained document is written in natural language without specific rules. There are no limitations in expressiveness on what can be specified in natural language. Further, it is a free text in natural language that can be understood by all parties, and no special training is required. On the other hand, disciplined documentation is a structured natural language, where the requirements engineer follow rules on how the statements should be written making the document organized. Requirement documentation include various types of statements, such as system requirements, software requirements and concept definitions.

In the ever-evolving field of software development, accurately interpreting and understanding of the software requirements is crucial to the successful deployment of software applications. However, software requirements often contain ambiguities, which can introduce unintended variability that must be identified and addressed [2]. Software requirements encompass both functional and nonfunctional aspects, which are fundamental in defining the system's behavior and performance [3], [4], [5]. These requirements provide a clearer understanding of the project and are foundational to defining the features of the project under development. The features include both FR and NFR, and the quality of these requirements impacts activities throughout the software development life cycle [6]. While the requirements themselves are

critical for product quality, their impact is often indirect, influencing the quality through the activities they govern.

FRs specify the functions or features a system must perform or support to meet user needs [7]. They describe what the system should do in terms of input, processing, and output. FR's include authentication of user, validation of data, generation of report and search functionality. On the other hand, NFR are referred to as "quality attributes" or "system qualities" which specifies the qualities or characteristics that the software system must possess [8]. NFR describes how well the system performs its functions rather than what those functions are. NFRs focus on aspects such as reliability, security, performance, usability, scalability and compliance. Unlike FRs, NFRs are often not directly observable by end-users but are critical for the overall success and acceptance of the system.

Traditionally, the classification of software requirements (FR and NFR) has been carried out through manual analysis by the engineers [9]. However, this process includes various challenges such as subjectivity, time consumption, and susceptibility to errors. It is a tedious task, prone to various errors, requiring a lot of effort, where each requirements document is needed to be read, analysed and classified manually. In response to the growing complexity of modern software projects and the ever-expanding volume of textual requirement documents, there is an increasing demand for automated and accurate techniques for requirement document classification [10], [11]. Requirement document challenges are very much in common with natural language documents challenges, such as semantic and syntactic ambiguity, synonymous nature and coherence [12]. In the context of interpreting software requirements, methodologies like NLP techniques have played a crucial role [13]. Additionally, there has been a substantial surge in the utilization of techniques such as deep learning (DL) and ML models for classifying software requirements [14], [15]. For requirement classification, we are leveraging the SmartNet dataset, sourced from industry, for evaluating the classification of FR and NFR. This dataset encapsulates real-world complexities such as-

- Emerging requirements- Customers' needs and preferences may change over a period of time, resulting in changing requirements. These requirements serve as inputs to the next stages in the software development life cycle for the planning of schedule [16]. To handle the evolving requirements, requires flexibility in the classification process to ensure that the system continues to meet the customer expectations [17].
- Varying stakeholder viewpoints - One of the most crucial aspects in the process of requirements definition is clarifying and establishing clear expectations of the client [18]. Different stakeholders, such as business analysts, end-users and technical experts, may have diverse perspectives on requirements [19]. Balancing these and ensuring that all stakeholders' needs are adequately addressed is important.
- Compatibility with current systems - Ensuring compatibility and coherence between new emerging and existing requirements can be complex and a careful analysis and classification of requirements is required [20].
- Insufficient customer requirements - Customers may provide incomplete requirements, making it difficult to classify them accurately [21].

In response to these challenges, we propose an innovative approach integrating NLP with an optimized CNN. Through this fusion, we aim to enhance the accuracy and efficacy of FR and NFR classification, thereby facilitating more robust and tailored software development processes. This paper presents a CNN framework for classifying the requirements automatically. The primary contributions of this research are summarized as follows:

1. Proposes an innovative integration of NLP techniques with CNN, augmented by the FPO algorithm, to automate the classification of software requirements into FR and NFR.
2. To perform supervised machine learning by applying precise and systematic labels to annotate the industry-standard SmartNet dataset
3. The effectiveness of the proposed approach is assessed using accuracy, precision, recall and F1-score, thus demonstrating the effectiveness of the CNN-FPO framework in accurately classifying requirements.

The rest of the paper is organized as follows: Section 2 reviews the related works on the software requirement classification. Section 3 discusses the proposed research methodology. Section 4 presents the experimental evaluation results and discussion, Section 5 discusses the internal and external validity threats and Section 6 concludes the paper with prominent research observations.

2 Related works

In recent years, there has been significant interest and a wide range of studies dedicated to developing innovative approaches for automated software requirement classification. Researchers have explored various techniques, including deep learning methods, ML algorithms, rule-based systems, and hybrid approaches. This section is organized into two subsections: a review of rule-based approaches followed by a review of ML approaches.

2.1 Rule- Based approaches

Over the past decade, rule-based approaches have attracted significant attention for classifying software requirements. These methods involve constructing models that use predefined rules and syntactic elements to categorize requirements based on their content. Goal and use case modeling is considered as an important approach for understanding the requirements. However, goals and use cases are often embedded within another content in requirements specifications documents. To address this, author in [22] has developed a new rule-based approach to automatically extract goal and use case models from natural language documents. The approach achieves 82% recall and 85% precision rates for extracting goal and usecase models. To reduce the manual analysis of natural language requirements for large open-source projects, in [23] author has proposed a rule based natural language technique. Initial results suggest that it reduces the effort required to analyze software requirements for open source projects. Author in [24], presented a proposal that automated the classification of natural language requirement sentences into NFR using a rule-based classification technique. It makes use of thematic roles to achieve this. Additionally, the proposal identifies the priority of extracted NFR sentences on the basis of their occurrence in multiple classes within the document. Further author in [25], has presented a framework for the automatic detection and classification of NFRs from natural language requirements. The approach involves parsing the natural language requirements to extract multiple features. The presence of specific combinations and relationships among these features is used to uniquely identify and classify the requirement as an NFR belonging to a particular category. The approach in [26], aimed to automate the detection of NFR using a text classifier enhanced with a part-of-speech (POS) tagger. The researchers identified nine groups of keywords, including adverbs, adjectives and model keywords. The frequency of each keyword was included as a feature in the main list of features and ranked using smoothed and non-smoothed probability measures. A threshold was set for each keyword to assign it to a specific NFR type. The results presented in the paper surpass recent studies in the field, achieving a higher accuracy of 98.56% using 10-fold cross-validation. Author in [27], has used an approach to classify stakeholders' quality concerns for requirements specifications. An iterative approach is used for training the classifier to classify NFR using dissimilar datasets. This approach is evaluated against natural language documents received from Automotive Organization and Siemens Logistics. Results show that though it is not able to classify all the NFRs, it is useful in any error-prone task. Similarly, author in [28] has presented an approach for fault classification using a noval systematic fuzzy approach. The model used the Decision Tree (DT) and a knowledge representation, for initial stage in classification, followed by a fuzzy rule based method for the final classification. Author in [29] proposed a hybrid approach that combines rule-based methods with ML techniques. The authors utilized extensive NLP techniques to convert unstructured requirement specifications into a corresponding goal model. Rule-based components, such as parts-of-speech tagging and dependency parsing, are employed to identify syntactic structures based on predefined rules. In contrast, the inclusion of contextual and synonymy vector generation, along with the FiBER transformer model, introduces ML elements into the framework.

Owing to the increasing complexity of software requirements documents and the expanding scope of software fields, rule-based approaches, which were effective, have become less prevalent. Researchers have diverted their focus to other statistical methods and ML algorithms, moving away from traditional rule-based methodologies.

2.2 Machine learning based approaches

In the field of software requirements, a significant number of studies have adopted machine learning approaches for identifying and classifying requirements. Early detection of requirements is crucial in the evaluation process that helps in building the initial design. Author in [30] propose a semi-supervised approach for text classification of NFR. Classification leverages a limited set of categorized requirements by utilizing the knowledge from uncategorized ones, along with specific textual properties. Results show that the approach has accuracy rates above 70%, higher than the one obtained using supervised methods. The author in [31], has represented a Search-Based Software Engineering (SBSE) for selecting a set of software requirements. A systematic review is presented that analyzed and categorized SBSE approaches to address software requirement selection problems and the current techniques used to address these problems. Morever in [32], author has proposed an approach to improve the classification of FR and NFR. Author has contributed an approach for preprocessing requirements that normalizes requirements before they are given for classification. Tera-PROMISE repository is used for the study. It is found that the preprocessing technique improved the performance of the existing classification methods. The automatic classification of software requirements from the text documents has gained huge significance among the researchers in recent times [33], [34], [35]. The author in [36] reviewed the application of different ML algorithm for classifying software requirements. Both SVM and KNN algorithms were used as classifiers for requirement classification. These algorithms were trained using a PROMISE_exp dataset which consists of labeled requirements. A bag of words (BoW) concept was employed for classification and results show that SVM performs better than KNN in terms of achieving a precision of 73%. Proper classification of security requirements present in the Software Requirement

Specification document has been problematic for the developers. In [37], author has proposed a method for classifying security requirements into four types (authentication, access control, cryptography-encryption and integrity of data) using J48 decision tree. The effectiveness of the prediction models is evaluated using requirement specifications collected from 15 projects developed by students at DePaul University. The author in [38], implemented different ML models such as SVM, SGD, and Random Forest models along with NLP for classifying review statements from an user app into FRs and NFR's. The models are trained on a public dataset namely PROMISE Software Engineering Repository dataset. Results show that the SVM model with TF-IDF exhibits excellent classification results. However, these models are time consuming and there is a need to investigate the application of models which have a faster execution time. Another author in [39] presented an approach for reducing misclassification rate and retrieving requirements for data-intensive applications. Author suggested to use Word Embedding followed by random forest classifier, as none of authors have used so far. With a precision of 0.89, F1 score of 0.91 and recall of 0.93, analysis showed that this approach produced a relatively high classification result for data-intensive systems. A comprehensive review of various algorithms for ML along with their applications in identifying and classifying NFR is presented in [40]. It provide insights into the effectiveness, strengths, and limitations of different machine learning approaches for handling NFR in software engineering. A supervised ML approach utilizing syntactic and lexical features is developed by author in [41]. Author employed over-and under-sampling strategies to handle imbalance in the data and the classifiers are cross-validated using performance metric like precision, recall, and F1 score. Experiments related to the support vector machine classifier on RE data challenge dataset. Author acheived the recall and precision for security and performance with ~92% precision and ~90% recall. A study to enhance the analysis of extracting 14 categories of NFRs from unconstrained requirements documents was conducted by authors [42]. They used 11 natural language documents from the "iTrust and PROMISE" datasets. Five different ML classifiers were compared for identifying NFRs. Their findings revealed that word vector representation combined with the SVM classifier was twice as effective as the Naive Bayes (NB) classifier. Additionally, the k-Nearest Neighbor (KNN) classifier with a distance matrix achieved an F1 measure score of 54% (precision and recall), while the NB reached upto only 32%. Authors in [43], has combine the N-Gram TF–IDF feature selection with binary and multiclass classifiers to automate the classification of refactorings, which is a process of improving the internal structure of the system without changing its quality. The model used about 2,867 commit messages extracted from open-source Java projects. Results indicate the F-1 score of the model reaches up to 90%. A systematic mapping study that surveys and synthesizes existing research on the application of natural language processing (NLP) techniques in requirements engineering is presented in [44]. It surveys the methodology of NLP4RE, to understand the current techniques and identify the gaps. It provides an overview of NLP approaches, methodologies, tools, and challenges in the context of requirements engineering.

Vectorization is a crucial process in NLP and semantic analysis that transforms text into numerical representations, which can be further used for classification. The author in [45], evaluated the impact of different vectorization methods. Author examined variety of methods: TF-IDF, Word2Vec and Doc2Vec. Four classifiers were used including Logistic Regression (LR), Naive Bayes (NB), and Random Forests. Experiments were conducted using the Tera-PROMISE repository, which contains 635 instances, including 370 NFR and 255 FR. The evaluation focused on four categories: operational, performance, security, and usability. The researchers found that both the Doc2Vec and SCDV vectorization methods outperformed traditional methods. In [46], author used a RE data challenge dataset for classifying FR and NFR. A word embedding technique was employed for extracting text related features and a fastText model was used for classifying FR and NFR. The performance of the fastText model is evaluated and it was observed that the model achieves phenomenal results with an excellent F1-score of 92.8%.

Multi-label classification methods for requirements documents involve techniques and approaches tailored to handle scenarios where a single requirement can belong to multiple categories. The study in [47] has explored the challenges associated with NFR. The objective is to reduce misclassification by aiding stakeholders in addressing NFRs during the early stages of development by automatically classifying requirements. For that author has proposed a multi-label requirement classifier based on CNNs that categorizes NFRs into five distinct classes: portability, efficiency, reliability, maintainability and usability. In another paper author has proposed a deep learning architecture for multi label classification in legal field [48]. Author has addressed an issue of good quality datasets labeled by humans, that restricts practitioners and researchers from gaining good performances and released a legal multi-label dataset for classification. The approach is evaluated on POSTURE50K and other multi-label dataset EUROLEX57K. It showed that the proposed methodology achieves better performances as compared to other four recent methods on both the legal datasets. Authors in [49], proposed a fuzzy similarity approach combined with K-Nearest Neighbors (FSKNN) for multi-label sentence classification. It involves identifying the k-nearest neighbors from each training pattern. In a separate study, in [50], author introduced an automated

system for identifying NFR using sentence-based classification algorithms of FSKNN. The result stated that the Semantic-FSKNN reduced the loss by 21.9%, and also raise the value of accuracy by 43.7%, as compared to FSKNN method A Bidirectional Gated Recurrent Neural Networks (BiGRU) is presented for classification of natural language requirements using raw text in [51]. It classifies requirements into functional and variety of labels of non-functional types with limited preprocessing technique to model the classification as a multilabel.

CNN are a type of deep learning models designed primarily for processing structured grid data such as images. In [52], authors presented a study on the automatic classification of requirements using CNNs. The research likely explores how CNNs can effectively classify software requirements into different categories, offering insights into the potential of deep learning techniques for requirement engineering tasks. In [53], authors examined the implementation of two ML models namely Artificial Neural Network (ANN) and CNN for classifying NFRs into different classes of requirements. Both ANN and CNN models were experimentally evaluated and it was observed that CNN is more effective in classifying NFR's as compared to ANN by achieving a precision of 94 % and recall of 97%. Authors in [54], deployed recurrent neural networks (RNN) and NLP for classifying NFR. The RNN model was developed and trained for processing the sequential natural language text and thereby classifying the NFR which is in the text form to different classes such as maintainability, operability, performance, security, and usability. The RNN model with NLP achieved a classification accuracy of 88% when tested for a diverse dataset consisting of one thousand NFR. An automatic requirement classification technique is presented in [55], which uses a graph attention network (GAT) known as DBGAT and a pre trained BERT model. The BERT model enhances the generalization of the model which is trained on the PROMISE dataset. The proposed DBGAT model achieves a higher classification performance in terms of F1 score of 91% (for known data) and 88% (for unknown data) with robust generalization capacity. A combination of NLP with AI-based techniques is presented in [56]. Author has proposed a method for classifying NFR in agile user stories. The framework is a combination of artificial intelligence and NLP techniques to automate the prediction of NFR. One seminal work that highlights the effectiveness of CNNs in text classification is presented in [57], which is used for sentence classification tasks. To understand the context it is important for grasping the word sense. However, feed forward network architecture of traditional CNN is insufficient to reflect this factor. For that, author has proposed a contextual CNN (C-CNN) by adding recurrent connection to the convolutional layer. In [58], authors reviewed the state-of-the-art CNN approaches, focusing from traditional models to deep learning models. A comparison of different techniques, with the pros and cons of different performance evaluation metrics are also provided. Similarly, the work in [59], proposes a hierarchical text classification approach based on CNN, aiming to capture hierarchical relationships between classes in text data. In another study, author has evaluated BERT's performance for inter- document and intra-document classification tasks. It is executed on a corpus of 1,303 requirements sources. BERT model is fine-tuned for functional classification where each requirement is classified as either functional or nonfunctional. When compared with a baseline model, Word2Vec, BERT model achieves higher classification accuracy [60]. Authors in [61] introduced a system named Norbert that fine-tunes BERT, a language mode leverages transfer learning techniques for requirements classification tasks. The critical task of classifying NFR using transfer learning models is presented by the authors in [62]. The process is evaluated using different transfer learning models, including BERT, Distil BERT, XLNet, Electra-base, Distil Roberta and Electra-small, for this purpose. In [63], author has proposed a method for early detection of NFR using NFR-classifier. The approach is used to detect and classify the quality concerns of the stakeholder across requirements specifications by using a noval iterative approach. This approach is evaluated against document obtained from logistics deparment of seimens and automotive organization.

A hybrid DL model is implemented in [64] for classifying the requirements. The hybrid model consisted of LSTM and bidirectional LSTM (BiLSTM) with ANN for classifying NFR. Results validate the effectiveness of the hybrid model when compared to the single LSTM and BiLSTM models. A hybrid approach known as FNReq-Net is presented in [65] for classifying FR and NFR. The model is tested and compared with other classifiers such as Naive Bayes, Logistic Regression, and SVM. As observed from the results, the hybrid FNReq-Net improves the classification performance by 4% for the PROMISE dataset.

The availability of datasets plays a crucial role in training and evaluating requirement classification models. To provide resources for researchers and practitioners in the field of requirements engineering author in [66] introduced a dataset named PURE, which contains public requirements documents. The dataset can be used for tasks such as training ML models, evaluating algorithms, and conducting empirical studies in requirements engineering.

Beyond recent advancements in deep learning architectures, which have improved the ability to capture complex patterns, several challenges and gaps remain. A key issue identified by the study is the shortage of labeled datasets for training. While commonly used datasets like PROMISE [67] and PURE offer some resources, there is a clear need for additional labeled data. Moreover, the performance of identical ML algorithms varies significantly across studies performing

well in some cases and poorly in others primarily due to differences in dataset quality and characteristics. Additionally, much of the literature lacks a systematic approach to feature identification and selection.

Building on insights from existing literature, this work proposes leveraging a CNN model, enhanced with the FPO algorithm and NLP techniques, to classify FR and NFR using an industry-standard dataset. The integration of FPO facilitates optimized feature subset selection, aiming to improve classification accuracy and model efficiency. Leveraging a manually annotated industry dataset ensures the approach remains grounded in real-world software engineering contexts, enhancing both its applicability and relevance.

Following Table 1 summarizes researchers work in the domain of requirement classification. Overall, the table provides a consolidated view of the research landscape in requirement classification. It helps in identifying commonly used classification methods, performance benchmarks, dataset trends, and existing challenges, thereby guiding future work in improving classification models, expanding datasets, or exploring under-researched areas.

Table 1: Summary of researcher's work in the field of requirement classification

Authors	Year of publication	Techniques used	Dataset characteristic	Performance metrics	Key findings/ Gaps
[22]	2015	Rule-based approach	PROMISE dataset is used consisting 625 requirements from 15 software projects.	Accuracy 88% Recall 82% Precision 85%	GUEST's artifact classifier surpasses Mallet and was more effective than Casamayor's classifier.
[23]	2011	Rule-based approach	SourceForge projects is used as our dataset. It allows access to 230,000 OSSD projects.	Precision 94% Recall 64% F-measure 76%	Need for refinement of parsing rules. Dependence on specific datasets and quality models.
[24]	2016	Rule-based approach	PROMISE corpus, Concordia RE corpus	F-measure 97%	Presence of misclassifications. Need for a larger corpus of quality requirements documents.
[25]	2014	Pattern-based rules for detecting and classifying NFRs	Promise repository	--	Creating new rules for ontology development.
[26]	2008	Automatic classification of requirements using text classifier	Corpus contained 15 SRS problem statements, a total of 765 sentences:65% of them are "NFR", while 35% of them are "FR"	Accuracy 98.56%	Builds an information retrieval technique by incorporating linguistic knowledge.
[27]	2007	Classification algorithm	30 requirements specifications document designed as term projects by students at DePaul University	--	The classifier struggles with detecting certain NFR types, such as "look-and-feel," indicating a gap to handle subjective language. Augmenting dataset with more examples of challenging NFR types.
[28]	2013	Systematic fuzzy rule based approach for fault classification	The dataset used here consists of fault current signal samples (in per unit values) obtained from a simulated	--	DT induced fuzzy rule-based approach provides improved accuracies compared to Heuristic fuzzy logic systems

			power system model		
[29]	2024	NLP-driven framework with FiBER transformer model	Promise_exp dataset	Accuracy 92%, Precision 92%, Recall 91% F-Score 92%	The framework has not been tested on agile requirements formats, such as user stories, which often contain unique linguistic structures.
[30]	2010	Semi-supervised approach for NFR identification and classification.	Standard collections of documents.	Accuracy rates above 70%	Semi-supervised requires less human intervention for requirement labeling than fully automated methods.
[31]	2015	Requirements selection and prioritization using search based approaches.	--	--	Requirements engineering is complex. This highlights the need for structured approaches to manage its multifaceted nature effectively.
[32]	2017	Automated classification of NFRs using ML algorithms	Tera-PROMISE repository	Precision 95% Recall 94% F-score 94%	Among the ML algorithms tested, Binarized Naïve Bayes (BNB) outperformed others. Challenges in differentiating usability requirements.
[33]	2019	Ensemble ML algorithms to classify FR and NFR	--	--	Experimental results show that gradient boosting outperforms random forest in terms of accuracy when classifying NFR. To explore other ML algorithms.
[34]	2022	NFRs prediction combined with a layered system architecture with ML and data visualization methods.	Dataset of NFRs was collected from 312 IT professionals and academicians, encompassing 17 attributes across 5,304 instances.	Accuracy 98.89%**.**	Proposed model outperforms existing techniques in data completeness, accuracy, prediction reliability and stability. Need for NFR Prioritization Mechanism.
[35]	2010	Review of the methods used for classification of document and text mining	Zoo dataset Forest Cover Type dataset	--	The study does not explore specific techniques beyond GSOM, leaving a gap in understanding how other methods could contribute to OLAP's analytical depth.
[36]	2021	Software requirements classification using ML	PROMISE_exp	F-measure 74%	To improve the accuracy and precision of the model, by modifying logistic regression algorithm.
[37]	2016	Text mining and	Dataset consists of documents collected	--	Does not focuses on other NFR such as performance,

		classification modeling of security requirements in SRS documents	from 15 projects developed by MS students at DePaul University. It consists of 326 NFRs specifications classified into 9 NFR types [37]		usability, and maintainability. Reliance on J48 Decision Tree Alone.
[38]	2022	ML based data modeling to identify requirements from review statements of user app	PROMISE dataset User app reviews dataset	Accuracy 83%.	To create new datasets comprising user app reviews and software requirements. XGBoost, CNN and RNN to be implemented.
[39]	2023	Word Embedding with a Random Forest Classifier to classify requirements	Banking dataset comprising 2812 requirement instances divided into 4 categories.	Precision 89% Recall 93% F1 score 91%	To increase the requirement dataset to mitigate the unbalance nature.
[40]	2019	Review of 24 ML methods for identifying and classifying NFRs	--	--	Lack of labeled datasets and a standard definition of NFRs are the open challenges.
[41]	2017	Classification of NFR using supervised ML	Quality attributes (NFR) dataset	Precision ~92% Recall ~90%	With automatic word feature selection, the classifier achieved higher recall for NFR classification but at the cost of lower precision.
[42]	2013	Extraction of NFRs in natural language documents using NLP.	NFRs documents like agreements, manuals, requirements documenets and user manuals containing NFRs categorized to 14 NFR categories	F1 measure 54%	Limited exploration of feature types beyond words exploring more advanced linguistic features that will enhance accuracy for NFR classification.
[43]	2021	Classification of refactored code	Self-affirmed-refactoring dataset containing 2867 commit messages	F-1 ~90%	To examine how well the approach applies to projects across various domains. To develop a tool that facilitates the detection of self-affirmed refactoring commits.
[44]	2021	Survey of NLP4RE	--	--	Lack focus on newer NLP advancements like transformer-based models like BERT. Lack of focus on evaluation metrics and benchmarking standards.
[45]	2018	Comparative analysis to	Tera-PROMISE Repository. The	--	Sparse Composite Document Vectors

		assess different vectorization and classification methods	repository contains 255 FRs and 370 NFRs.		(SCDV) and Doc2Vec achieve higher performances than traditional ones [45].
[46]	2020	Classification of FR and NFR using Word2vec and fast Text	625 requirement text from International Requirements Engineering Conference's 2017 Data challenge dataset.	Precision 92.8% Recall 92.8% F1 score 92.8%	The study mentions future work with BERT, and other advanced NLP models like GPT or RoBERTa, which might offer valuable insights.
[47]	2020	Requirement classification using CNN technique to minimize misclassificati on of NFRs	Manual labeled corpus of NFRs from SRS documents.	--	The technique to be implemented on balanced corpus to improve the results.
[48]	2022	Deep learning architecture for multi-label document classification.	POSTURE50K, a novel legal multi-label classification dataset containing 50,000 legal opinions. EUROLEX57K multi-label dataset.	Precision 87.8% Recall 80% F1 score 82%	Limited exploration of data augmentation techniques. Comparison with domain-specific transformers beyond RoBERTa. Challenges in handling labels with very few training samples.
[49]	2012	Classification using fuzzy similarity measure (FSM) and KNN.	Reuters-21578, RCV1, and 20 Newsgroups.	F1 score 97.27% microaveraged breakeven point (BEP) 97.27%	Although FSKNN reduces computational cost by clustering training patterns, the KNN-based approach still face scalability challenges on large datasets.
[50]	2015	Classification of NFR Using Semantic-FSKNN	Dataset contains 1342 sentences from six different datasets	Precision 73.9%	Semantic-FSKNN method reduces the error rate by 21.9%, and also raises the accuracy by 43.7%.
[51]	2022	Bidirectional Gated Recurrent Neural Networks (BiGRU) to classifiy FR and NFR	PROMISE dataset EHR dataset	--	Requirements can be effectively classified into FR and NFR using the presented RNN based deep learning system, with minimal text prepossessing and no feature engineering.
[52]	2016	Classify FR and NFR using CNN	DOORS document database	precision 73% Recall 89%	To increase training data to improve the accuracy. Consider documents from other domains as well. To provide the user with the explanantion as why a content element is classified incorrect.
[53]	2019	Classify software requirements	International Requirements Engineering	Precision 82% and 94% Recall 76% and	Investigating the feasibility of other ML techniques. To design

		using ML techniques	Conference's 2017 Data challenge dataset. Promise dataset.	97% F1-score 82% and 92%	fully automated ML based system. Use of BERT to improve the performance.
[54]	2021	RNN to classify NFRs	International Requirements Engineering Conference's 2017 Data challenge dataset. Promise dataset.	Accuracy 88% Precision 84% Recall 85% F1-score 84%	In future to include FRs to assess the model's performance.
[55]	2022	Requirement classification using BERT and graph attention network (GAT)	PROMISE dataset	F1-score 91%	Work on dynamic graph research in the future. Expand the data volume through data annotation to improve model capability.
[56]	2023	NLP and AI technique to predict NFR in user stories	--	--	It mitigates the reported causes of the problems in the literature: limited documentation and the functionality-focused behavior of agile requirements.
[57]	2018	Contextual-CNN for sentence classification	Stanford Sentiment Treebank and question categorization dataset TREC Sentiment prediction dataset	Accuracy 95.2%	Combining recurrent connections with convolutional layer, the network effectively integrated feature extraction and context modulation.
[58]	2020	Survey on text classification	--	--	Challenges related to the quality, structure, and availability of the data used to train, validate, and test ML model.
[59]	2020	Heirarchical text classification using Deep learning	Dataset consist of 51,325 sentences from 8 online news websites belonging to technology, sports and entertainment	Accuracy 94.02% Precision 92.59% Recall 88.12%	To develop Bi-LSTM based approach to improve hierarchical structure. To optimize the model as it takes long training time.
[60]	2024	Deep NN for classifying requirements	Dataset of 1303 requirements sourced from five mechanical design documents.	Matthews correlation coefficient (MCC) 0.95	While BERT is used, the study does not explore embeddings from other advanced transformer models, such as MPNet and RoBERTa.
[61]	2020	Transfer learning for requirement classification.	Promise NFR dataset	F1 score 92%	To analyze the performance of language models such as XLNet or XLM. To investigate how a multiclass multilabel FR classification would perform.
[62]	2023	NFR classification using Transfer learning	Dataset consist of aproximately 1445 records with 12 classes of quality	Accuracy 91.48% Precision 91.48% Recall 91.48% F1 Score 91.48%	Small dataset size, which affect generalizability. Explore ensemble methods and other transfer

			attributes		learning strategies to enhance NFR classification.
[63]	2006	Early detection of NFR	15 Requirements specifications developed by MS students at DePaul University. Specification document obtained from Siemens Logistics and Automotive Organization.	--	Lacks insights into how often the model should be retrained to maintain accuracy across different contexts. Testing on larger datasets could improve robustness and generalizability.
[64]	2023	Hybrid deep learning for NFR classification	Dataset containing 1000 NFRs	--	Hybrid models demonstrated superior performance in terms of precision, recall, and F-1 score.
[65]	2016	Hybrid predictor namely FNReq-Net for classification	Promise dataset Promise-exp dataset	F1-Score improved by 4% and 1%	To explore different word embeddings methods. Optimize the hyperparameters of traditional machine learning classifiers using a genetic algorithm.
[66]	2017	Creation and analysis of a natural language requirements dataset.	PURE (PUblic REquirements dataset), a dataset of 79 publicly available natural language requirements documents. Dataset includes 34,268 sentences.	--	The future work involves converting the entire dataset to XML format and enhancing it by incorporating additional documents. To expect RE researchers annotate the requirements for specific tasks.
[67]	2017	Fault prediction using ML approach	Promise dataset	--	SVM and Random Forest algorithm can be used as a recommendation ML algorithm for fault prediction.

3 Proposed research methodology

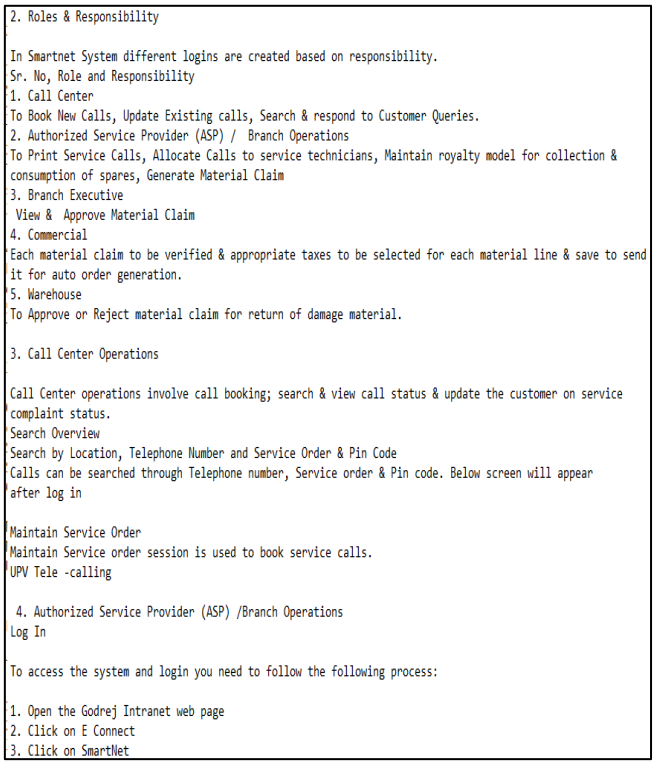
This research focuses on classifying FR and NFR for an industry SmartNet dataset to facilitate successful deployment of the software. Feature selection is optimized using the FPO optimizer that selects an optimal set of features. A CNN model is then employed for classification
based on the optimal set of features. This process is complemented by NLP techniques to manage the sequential data. The stages involved in implementing the proposed approach are outlined as follows:

3.1 SmartNet dataset manual annotation

For this research work, an industry SmartNet dataset [68] is used given by Godrej as shown in Figure 1. SmartNet is designed to handle the services provided to the customer upon delivering the product. It is designed to improve service functionality & customer satisfaction. SmartNet dataset includes unconstrained requirements with unlabeled sentences. It contains many real world complexities as described in Section 1. A sentence level manual annotation is necessary for conducting supervised learning experiments.

To perform the process of manual annotation, firstly all the requirements sentences from the document are extracted. It is done so by using sentence boundary, where sentence boundary are identified by capital letter and punctuation marks. In most of the cases in the requirement document, every sentence talks about one software requirement. But it is not mandatory that all the parsed sentences are requirement sentences. So, the annotation process is performed at sentence level. Once all the requirements are extracted, they are labeled by the group of experts who have in-depth knowledge of the data. The annotation process was carried out in an offline mode by the experts. The interpretation of functional and non-functional requirements differ across industries,

influencing how results are perceived and applied. To avoid this, domain experts included software engineers from Godrej that ensured the labeling process is aligned with industry standards. For each sentence three options were provided namely- FR, NFR and others. After the experts read the sentences, they had to decide if the sentence was describing a FR, NFR or something else. If the expert finds a sentence which doesn't fit in any of the requirement category, then the others option could be selected.



2. Roles & Responsibility

In Smartnet System different logins are created based on responsibility.
Sr. No, Role and Responsibility
1. Call Center
To Book New Calls, Update Existing calls, Search & respond to Customer Queries.
2. Authorized Service Provider (ASP) / Branch Operations
To Print Service Calls, Allocate Calls to service technicians, Maintain royalty model for collection & consumption of spares, Generate Material Claim
3. Branch Executive
View & Approve Material Claim
4. Commercial
Each material claim to be verified & appropriate taxes to be selected for each material line & save to send it for auto order generation.
5. Warehouse
To Approve or Reject material claim for return of damage material.

3. Call Center Operations

Call Center operations involve call booking; search & view call status & update the customer on service complaint status.
Search Overview
Search by Location, Telephone Number and Service Order & Pin Code
Calls can be searched through Telephone number, Service order & Pin code. Below screen will appear after log in

Maintain Service Order
Maintain Service order session is used to book service calls.
UPV Tele -calling

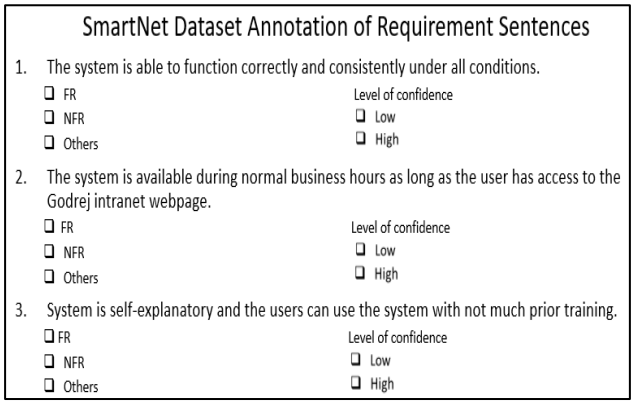
4. Authorized Service Provider (ASP) /Branch Operations
Log In

To access the system and login you need to follow the following process:

1. Open the Godrej Intranet web page
2. Click on E Connect
3. Click on SmartNet

Figure 1: SmartNet dataset sample sentences without annotation

After labeling the requirement sentence, the expert had to express his confidence for each sentence by selecting two levels of confidence: low level or high level for each answer as shown in Figure 2. To avoid annotation error, each sentence had to be labeled by all the experts. To ensure the accuracy of the labeling process, we followed a specific set of criteria for accepting each review. Firstly, for a review to be considered as valid, it must be evaluated and completed by all the experts. Secondly, if the experts categorize the review differently, the category with the higher confidence level is selected. Lastly, if both the confidence levels are equal, preference is given to the reviewer having more experience.



SmartNet Dataset Annotation of Requirement Sentences

1. The system is able to function correctly and consistently under all conditions.
   - ❑ FR
   - ❑ NFR
   - ❑ Others

   Level of confidence
   - ❑ Low
   - ❑ High

2. The system is available during normal business hours as long as the user has access to the Godrej intranet webpage.
   - ❑ FR
   - ❑ NFR
   - ❑ Others

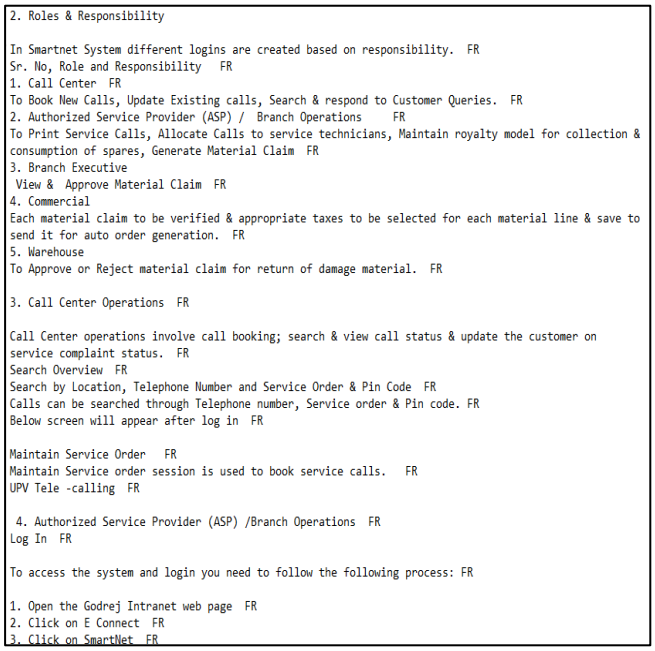
   Level of confidence
   - ❑ Low
   - ❑ High

3. System is self-explanatory and the users can use the system with not much prior training.
   - ❑ FR
   - ❑ NFR
   - ❑ Others

   Level of confidence
   - ❑ Low
   - ❑ High

Figure 2: Sample of requirement sentences for manual annotation

The labelling process took over 2-3 days for the initial setup which included training to the software engineers and guideline creation, 1-2 weeks for annotating the document followed by quality control for 2-4 days to ensure consistency check and accuracy. The discrepancies found in the quality control process were resolved by discussing with the annotators and domain experts. Following Figure 3. shows sample of annotated dataset. After the annotation task is completed, we got a total of 217 requirement sentences, labeled by 10 software engineer experts with different levels of experience.



2. Roles & Responsibility

In Smartnet System different logins are created based on responsibility. FR
Sr. No, Role and Responsibility FR
1. Call Center FR
To Book New Calls, Update Existing calls, Search & respond to Customer Queries. FR
2. Authorized Service Provider (ASP) / Branch Operations FR
To Print Service Calls, Allocate Calls to service technicians, Maintain royalty model for collection & consumption of spares, Generate Material Claim FR
3. Branch Executive
View & Approve Material Claim FR
4. Commercial
Each material claim to be verified & appropriate taxes to be selected for each material line & save to send it for auto order generation. FR
5. Warehouse
To Approve or Reject material claim for return of damage material. FR

3. Call Center Operations FR

Call Center operations involve call booking; search & view call status & update the customer on service complaint status. FR
Search Overview FR
Search by Location, Telephone Number and Service Order & Pin Code FR
Calls can be searched through Telephone number, Service order & Pin code. FR
Below screen will appear after log in FR

Maintain Service Order FR
Maintain Service order session is used to book service calls. FR
UPV Tele -calling FR

4. Authorized Service Provider (ASP) /Branch Operations FR
Log In FR

To access the system and login you need to follow the following process: FR

1. Open the Godrej Intranet web page FR
2. Click on E Connect FR
3. Click on SmartNet FR

Figure 3: SmartNet dataset sample sentences with annotation

Table 2: Classification of sentences

	Sentences
FR	198
NFR	14
Irrelevant	5
Total	217

The distribution of these sentences are as shown in Table 2.

3.2 Text extraction

Text extraction is the process of retrieving text from a file (in this case, a .txt file) and making it accessible within a program. Here, the test data stored in the 'output1.txt' file is extracted into the variable 'text,' allowing it to be read, analyzed, and processed further. Following Figure 4 and Figure 5. shows the code snippet and output for text extraction process respectively.

```
f = open('output1.txt', 'r',encoding='utf8')
text=f.read()
f.close()
```

Figure 4: Text Extraction code snippet

```
text

' \n 1. Introduction  \n \nSmartnet system is designed to handle after Sales Service Processes of Security Solutions \nDivisio
n. This System will overcome day to day problems faced in BaaN operations. Smartnet \nsystem is designed to improve service fun
ctionality & customer satisfaction.  \nIt enables processes in association with call centers, dealers and Authorized Service Pr
ovides \n(ASP). The system is able to function correctly and consistently under all conditions.The system is available during n
ormal business hours as long as the user has access to the Godrej Intranet webpage.  \nSystem is self-explanatory and the users
can use the system with not much prior training. \nSystem is expected to resolve queries of around 10000 customers and this num
ber is expected to grow 5 times within the next year.\nIt is equipped with following  functionalities  \n\uf0a7 Call planning
\n\uf0a7 ASP Technician planning  \n\uf0a7 Call Processing  \n\uf0a7 Claim Handling  \n\uf0a7 Customer Satisfaction Management
\n\uf0a7 Depot Repair  \n\uf0a7 Warranty & Contracts Management  \n\uf0a7 Product Replacement Approval (PRA)  \n\uf0a7 ACT Proc
ess  \n\uf0a7 Franchisee (ASP) Payment  \n\uf0a7 Dealer Damage Handling  \n\uf0a7 SSD Management  \n\uf0a7 ASP Scheme Managemen
t  \n\uf0a7 SIS Management  \n\uf0a7 Institutional Customer Process  \n \nVarious transactions and masters are integrated with
Baan IV such as item data. Order types, \nsales order etc.  \n  \n2. Roles & Responsibility  \n \nIn Smartnet System different
logins are created based on responsibility.  \nSr. No, Role and Responsibility  \n1. Call Center  \nTo Book New Calls, Update
```

Figure 5**:** Text Extraction output

3.3 Text preprocessing

Text preprocessing is performed on the dataset to clean the data by removing the uncertainties present and thereby making the data suitable for the classification purpose. Preprocessing helps in extracting relevant and meaningful features from the data.

Following Figure 6. represents the block diagram for text preprocessing which is described in detail in the following subsections.

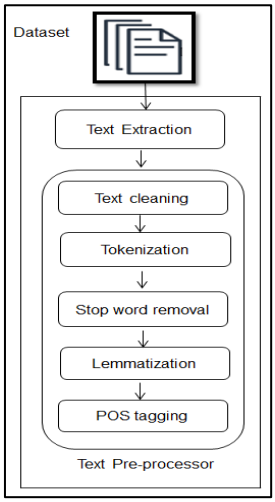


Figure 6: Block diagram of Text Pre-processor

3.3.1 Text cleaning

In this stage, after text extraction, the text undergoes cleaning to remove empty spaces and special characters. This involves converting the text to lowercase, replacing special characters with spaces, and removing non-word characters (anything other than letters, digits, and underscores). Additionally, any links or website references, as well as punctuation, are removed. This cleaning process produces text that is more manageable for further processing tasks. Following Figure 7. shows the output after cleaning the text.

```
#Text cleaning
def clean_word(text):
    text = text.lower()    #we want to the convert our text into lower case
    text = re.sub('\[.*?\]', '', text)  #substitute these spacial character into empty space
    text = re.sub("\\W"," ",text)    #if any non-word character present it replaces that
    text = re.sub('https?://\S+|www\.\S+', '', text)  #it removes any google link or website,it replaced
    text = re.sub('<.*?>+', '', text)  #these special cahracter also replaced with empty space..so these character will be remo
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)  #other than the above special character present in our text
    text = re.sub('\n', '', text)  #if any \n means new line.it will replace
    text = re.sub('\w*\d\w*', '', text)    #if any number,non-word present in our text,it will replace
    text = text.translate(str.maketrans(' ',' ',string.punctuation))
    #will consider only alphabets and numerics
    text = re.sub('[^a-zA-Z]',' ',text)
    #will replace newline with space
    text = re.sub("\n"," ",text)
    # will split and join the words
    text=' '.join(text.split())
    return text

clean_text=clean_word(text)
clean_text

'introduction smartnet system is designed to handle after sales service processes of security solutions division this system wi
ll overcome day to day problems faced in baan operations smartnet system is designed to improve service functionality customer
satisfaction it enables processes in association with call centers dealers and authorized service provides asp the system is ab
le to function correctly and consistently under all conditions the system is available during normal business hours as long as
the user has access to the godrej intranet webpage system is self explanatory and the users can use the system with not much pr
ior training system is expected to resolve queries of around customers and this number is expected to grow times within the nex
t year it is equipped with following functionalities call planning asp technician planning call processing claim handling custo
mer satisfaction management depot repair warranty contracts management product replacement approval pra act process franchisee
asp payment dealer damage handling ssd management asp scheme management sis management institutional customer process various t
```

Figure 7**:** Sample of text cleaning output

3.3.2 Tokenization

In the process of tokenization, text is divided into smaller paragraphs and then into individual sentences [69]. Then each individual sentence is divided into multiple small words or tokens. These tokens can be words, phrases, symbols, or other meaningful elements, depending on the type of tokenization and the language. A new sentence is identified by the capital letter at the start and a full stop or a question mark or an exclamation mark at the end of the sentence. A tokenization function 'word_tokenize' from NLTK, a python library is used for tokenization which helps in extracting english sentences from the document[70]. It is a foundational step in NLP and ML tasks, where it's essential for transforming unstructured text into structured data for further analysis

For example, when applying tokenization on "To print the service job sheets click on print service job sheet", the result will be a set of tokens: {To, print, the, service, job, sheets, click, on, print, service, job, sheet}. Following Figure 8. Shows the sample output for tokenization.

```
#Tokenization
from nltk import word_tokenize
words = word_tokenize(clean_text)
print(words)
```

```
['introduction', 'smartnet', 'system', 'is', 'designed', 'to', 'handle', 'after', 'sales', 'service', 'processes', 'of', 'sec
urity', 'solutions', 'division', 'this', 'system', 'will', 'overcome', 'day', 'to', 'day', 'problems', 'faced', 'in', 'baan',
'operations', 'smartnet', 'system', 'is', 'designed', 'to', 'improve', 'service', 'functionality', 'customer', 'satisfactio
n', 'it', 'enables', 'processes', 'in', 'association', 'with', 'call', 'centers', 'dealers', 'and', 'authorized', 'service',
'provides', 'asp', 'the', 'system', 'is', 'able', 'to', 'function', 'correctly', 'and', 'consistently', 'under', 'all', 'cond
itions', 'the', 'system', 'is', 'available', 'during', 'normal', 'business', 'hours', 'as', 'long', 'as', 'the', 'user', 'ha
s', 'access', 'to', 'the', 'godrej', 'intranet', 'webpage', 'system', 'is', 'self', 'explanatory', 'and', 'the', 'users', 'ca
n', 'use', 'the', 'system', 'with', 'not', 'much', 'prior', 'training', 'system', 'is', 'expected', 'to', 'resolve', 'querie
s', 'of', 'around', 'customers', 'and', 'this', 'number', 'is', 'expected', 'to', 'grow', 'times', 'within', 'the', 'next',
'year', 'it', 'is', 'equipped', 'with', 'following', 'functionalities', 'call', 'planning', 'asp', 'technician', 'planning',
'call', 'processing', 'claim', 'handling', 'customer', 'satisfaction', 'management', 'depot', 'repair', 'warranty', 'contract
s', 'management', 'product', 'replacement', 'approval', 'pra', 'act', 'process', 'franchisee', 'asp', 'payment', 'dealer', 'd
```

Figure 8**:** Sample output of tokenization

3.3.3 Stop words initialization and removal

Stop words are common words that hold minimal value or relevance. Removing them does not impact the meaning of the sentence. In the context of NLP, stop word initialization refers to the process of identifying and creating a list of stopwords that are removed while analyzing the natural text. Following Figure 9. shows the sample output of stopword removal as filtered_words.

```
#Remove stop words from our text
filtered_words = [word for word in words if word not in stop_words]
print(filtered_words)
```

```
['introduction', 'smartnet', 'system', 'designed', 'handle', 'sales', 'service', 'processes', 'security', 'solutions', 'divisio
n', 'system', 'overcome', 'day', 'day', 'problems', 'faced', 'baan', 'operations', 'smartnet', 'system', 'designed', 'improve',
'service', 'functionality', 'customer', 'satisfaction', 'enables', 'processes', 'association', 'call', 'centers', 'dealers', 'a
uthorized', 'service', 'provides', 'asp', 'system', 'able', 'function', 'correctly', 'consistently', 'conditions', 'system', 'a
vailable', 'normal', 'business', 'hours', 'long', 'user', 'access', 'godrej', 'intranet', 'webpage', 'system', 'self', 'explana
tory', 'users', 'use', 'system', 'much', 'prior', 'training', 'system', 'expected', 'resolve', 'queries', 'around', 'customer
s', 'number', 'expected', 'grow', 'times', 'within', 'next', 'year', 'equipped', 'following', 'functionalities', 'call', 'plann
ing', 'asp', 'technician', 'planning', 'call', 'processing', 'claim', 'handling', 'customer', 'satisfaction', 'management', 'de
pot', 'repair', 'warranty', 'contracts', 'management', 'product', 'replacement', 'approval', 'pra', 'act', 'process', 'franchis
ee', 'asp', 'payment', 'dealer', 'damage', 'handling', 'ssd', 'management', 'asp', 'scheme', 'management', 'sis', 'management',
'institutional', 'customer', 'process', 'various', 'transactions', 'masters', 'integrated', 'baan', 'iv', 'item', 'data', 'orde
```

Figure 9**:** Sample output of stopword removal

3.3.4 Lemmatization

Following stemming, the next step is lemmatization, where words are reduced to their dictionary form. Unlike stemming, which merely removes suffixes or prefixes and may produce incorrectly spelled words, lemmatization takes into account the word's meaning and context to produce accurate root forms, thereby resulting in a valid word form. [71]. The WordNetLemmatizer function from NLTK library in python is used to reduce words to their base form. Following Figure 10. shows the sample output for lemmatization performed on filtered_words and writes the output to a file named example.txt.

```
#Apply Lemmatization in text
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
lemmatised = [wordnet_lemmatizer.lemmatize(word) for word in filtered_words]
print(lemmatised)
with open("example.txt", "w") as f:
    for word in lemmatised:
        f.write(word+'    ')
```

```
['introduction', 'smartnet', 'system', 'designed', 'handle', 'sale', 'service', 'process', 'security', 'solution', 'division',
'system', 'overcome', 'day', 'day', 'problem', 'faced', 'baan', 'operation', 'smartnet', 'system', 'designed', 'improve', 'serv
ice', 'functionality', 'customer', 'satisfaction', 'enables', 'process', 'association', 'call', 'center', 'dealer', 'authorize
d', 'service', 'provides', 'asp', 'system', 'able', 'function', 'correctly', 'consistently', 'condition', 'system', 'availabl
e', 'normal', 'business', 'hour', 'long', 'user', 'access', 'godrej', 'intranet', 'webpage', 'system', 'self', 'explanatory',
'user', 'use', 'system', 'much', 'prior', 'training', 'system', 'expected', 'resolve', 'query', 'around', 'customer', 'number',
'expected', 'grow', 'time', 'within', 'next', 'year', 'equipped', 'following', 'functionality', 'call', 'planning', 'asp', 'tec
hnician', 'planning', 'call', 'processing', 'claim', 'handling', 'customer', 'satisfaction', 'management', 'depot', 'repair',
'warranty', 'contract', 'management', 'product', 'replacement', 'approval', 'pra', 'act', 'process', 'franchisee', 'asp', 'paym
ent', 'dealer', 'damage', 'handling', 'ssd', 'management', 'asp', 'scheme', 'management', 'si', 'management', 'institutional',
'customer', 'process', 'various', 'transaction', 'master', 'integrated', 'baan', 'iv', 'item', 'data', 'order', 'type', 'sale',
```

Figure 10**:** Sample output of Lemmatization

Choosing lemmatization over stemming for research improved the quality of feature extraction. Stemming reduced words to base form giving incorrect words like "secur", "servic", "handl", "oper" , "condit" etc. which hindered the process of feature extraction. On the contrary, lemmatization reduced words to their dictionary form giving meaningful words like "security", "service", "handle", "operation", "condition" therby improving the quality of feature extraction.

3.4 Feature extraction and selection

The important features from the text are extracted using two feature extraction techniques namely Bag of words (BoW) and Word2vec. Further feature selection is done using ChiSquare method which are discussed in the following subsections.

3.4.1 Feature extraction

This is a crucial step in classifying software requirements. Given the large volume of data in software requirements, it's vital to extract only the relevant features from the dataset. The extracted features are then used to identify the requirements using the proposed CNN model. In this research, for feature extraction two techniques are used namely BoW (Categorical word representation) and Word2Vec (Continuous word representation). Word2Vec captures semantic similarity between the words, making it suitable for tasks requiring nuanced understanding of relationships [72]. BoW, on the other hand, provides a straightforward representation of text documents and is computationally efficient and versatile. BoW is easily interpretable and applicable to a wide range of NLP tasks [73]. In this paper, classification is done using BoW and compared with Word2Vec classification as discussed in Section 4.2, to provide a comprehensive representation of text data. The process involved in the

BoW concept is as follows:

- The process of BoW begins with text tokenization in which the text is divided into individual words or tokens.
- After tokenization, a vocabulary is created by collecting all the unique tokens from the entire corpus where each unique token is assigned a unique index. A feature vector is created equal to the size of the vocabulary. The values in the feature vector are binary values to indicate the presence (1) or the absence (0) of the word. For every document, the number of occurrences of each word in the vocabulary is counted representing the frequency of that word.

For BoW model, the output of the lemmatization is given as input text which is transformed into a fixed-length vector. This is done by counting the number of times the word is present in a document. The BoW model provides the unique features related to the text present in the document. A data frame is created which displays the value '1' if the particular word is present. Else it displays 0. BoW do not capture syntactic information meaning the relationship between the words as they do not consider the order of the words [73]. Following Figure 11. shows how requirement sentence is represented as BoW vector

Requirement sentence- "The system is designed to handle after Sales Service Processes of Security Solutions". After Pre-processing the requirement sentence becomes as follows: ['system','designed','handle','sale','service','process', 'security', 'solution']

system	designed	handle	sale	service	process	security	solution
1	1	1	1	1	1	1	1

Figure 11: BoW feature representation

Word2Vec is a word embedding model used to convert feature words into dense vector representations also known as word embeddings. The embeddings determine the semantic relationship between the words that helps in classifying the natural text [72]. In Word2Vec model, words which have similar meanings are arranged close to each other and provide the contextual information. Word2Vec technique are of two types namely Continuous Bag of Words (CBOW) and Skip-gram [74].

Word2Vec operates based on the sliding window framework, which when applied over a sentence, for each word, it considers a content window of adjacent words. In this paper skip gram model is used where, given the target words, the context words are predicted. Word2Vec uses a 2-layer neural network to represent the word vectors. During training, the weights of the network are adjusted to minimize the prediction error. Once the model is trained, the hidden layer weights are considered as the word embeddings. These embeddings represents meaningful representations of words in a continuous vector space. These embeddings are then given as input to the classifier. Following Figure 12. represents Word2Vec embedding for word approve from the above requirement sentence.

```
model.wv['reject']

array([ 6.1788400e-05,  2.4389235e-04, -6.4859662e-04, ...,
       -1.7808526e-04,  7.3195819e-04, -1.9369277e-04], dtype=float32)
```

Figure 12: Word2Vec Embedding for the word reject

3.4.2 Feature selection

Feature selection is referred as a process of identifying and selecting the most relevant features from a dataset that contribute significantly to the classification tasks. For feature selection Chi Square model is used, where the output of feature extraction process is given as an input to Chi-square for finding the scores. Based on the scores, the words (features) are selected. Chi Square method employs a statistical method for selecting important and relavent features while managing discrete data [75]. Technically, the Chi Square method uses a contingency table to determine the independence between two categorical variables. In this process, features that are correlated with the target variables are selected for classification. A null (H0) and alternative (H1) hypothesis are used to make assumptions, which are represented as follows:

H0 :- There is no relationship between categorical feature and target variable

H1 :- There is certain relationship between categorical feature and target variable

In Chi square feature selection, categorical features represent requirements and the target variable is the variable you're trying to predict. H0 is the null hypothesis and H1 represents the alternate hypothesis. If the p-value is $\geq$ 0.05, we reject the alternative hypothesis (H1), indicating that the target variable and categorical features are not significantly related. Conversely, if the p-value < 0.05, we reject the null hypothesis, suggesting a significant relationship between the target variable and categorical features. Features with a p-value greater than 0.05 are excluded from the feature set, and the remaining features are retained for the classification process.

A contingency table is created based on the selected features, representing the frequency distribution of two categorical variables. This table displays the counts of observations within the different categories of the two variables. Chi-square selects 27 features for BoW vectorizer and selects 401 features for Word2Vec as shown in Table 3.

Table 3: Feature selection using Chi-square for BoW and Word2vec vectorizer

	BoW	Word2Vec
Chi-square	27	401

3.5 Classification using ML classifiers

Classification is a core task in ML aimed at assigning input data to predefined categories. Machine learning classifiers achieve this by identifying patterns from a dataset during training and applying these learned patterns to predict labels for new data. Three supervised ML algorithms are utilized: SVM, KNN, and CNN. SVM and KNN are selected based on their strong performance in similar studies in the literature [9]. CNN, a deep learning model and considered state-of-the-art in classification, is also employed. Details of these classifiers are discussed in the following subsection.

3.5.1 SVM classifier

SVM is a discriminative method used for classification in the literature [76]. the SVM classifier addresses the nonlinear classification problem by utilizing a "hyperplane." Feature vectors of requirement sentences are mapped to a high-dimensional space where each dimension can be linearly separated by a decision boundary, known as the hyperplane. Given the two classes, FR and NFR, a binary SVM classifier is employed. For enhanced classification performance, the SVM classifier parameters are configured to use a polynomial kernel (kernel=poly) with a degree of 10 and a maximum of 100 iterations (max_iter=100).

3.5.2 KNN classifier

KNN classifier [77] is a simple, intuitive, and widely used ML algorithm for classification. To classify an unknown data point, KNN calculates the distance between the unknown point and all the points in the training set by using an appropriate distance metric.

After calculating the distances, the new data point is assigned to the class that is very much common among its nearest neighbors through majority voting. After fine tuning, the parameters used were : [n_neighbors=3, p: 2]. p=2 corresponds to the Euclidean distance.

3.5.3 CNN classifier

Convolutional Neural Networks [52] have been widely used for the task of requirement classification. Initially, for requirement classification, the sentences are segmented into tokens (word). Then for each word a vector is generated using feature extraction techniques-BoW and Word2Vec. After feature extraction, feature selection is done using chi-square after which the vectors are passed through different layers in the CNN that are discussed in the following section:

- Convolutional Layers: The convolutional layers are the fundamental building blocks of the CNN architecture. A dot product is calculated between each filter and patches of the input data, allowing the model to extract relevant text features. The input to the convolutional layer has two dimensions: (Xtrain.shape[1], 1), where Xtrain.shape[1] represents the length of the sequence or the number of features per sample, and 1 represents the number of channels. Following Figure 13. shows Xtrain data input to CNN where each row represents the vector of each word and column represents the features. Then Xtrain.shape[1] would be n, representing the number of features dimensions for each word.

Words	Feature 1	Feature 2	...	Feature n-1	Feature n
system	473.6	-438.7	...	82.01	22.45
designed	127.6	-17.91	...	21.29	74.94
handle	327.3	-234.6	...	35.37	88.32
sale	228.9	947.1	...	-27.18	27.38

Figure 13: Sample represention of Xtrain data

- Activation functions: These functions are applied to the output layers of the CNN model. In this research a ReLU (Rectified Linear Unit) activation function is applied for improving the non-linearity of the model.
- Pooling layers: These layers are used to down-sample the size of the text.
- Dense layers: The dense layers are the fully connected layers of the network. The output of the final pooling layer is concatenated into one vector and feed to a dense layer. It takes in the features and transform them to classify the input requirements. In this research for binary classification, sigmoid activation function is used. The output layer represents neurons where each neuron represents a label belonging to one of the target categories i.e FR or NFR.

3.6 CNN model optimization using flower pollination optimizer algorithm

CNN is a fundamental classifier which has gained huge success in the requirement classification tasks. Optimizing CNN models enhance their performance, efficiency and scalability in requirement classification tasks [78]. In this research, a FPO algorithm is used for optimizing the CNN model for obtaining an optimal output.

FPO algorithm is a nature-inspired optimization algorithm used to solve optimization and search problems [79]. In optimizing CNN, the challenge lies in finding the most effective feature set to minimize loss and improve classification performance. FPO algorithm is instrumental in this process, as it searches for an optimal set of features that enhances the CNN model's overall

effectiveness. FPO algorithm is inspired by the pollination process of flowers carried out by bees to find optimal solutions in multidimensional search spaces [80]. The process involved in the FPO algorithm is discussed below:

- Initialization: FPO algorithm starts by initializing a population of potential solutions for the problem, with each solution represented as a point within a multidimensional search space.
- Objective Function: The optimization problem is guided by an objective function that assigns a fitness or cost value to each solution within the population. Each solution is assessed based on this fitness function, which indicates how effectively it addresses the optimization challenge. The aim is to locate the optimal solution, or the "fittest flower," in the population. The objective function is defined as follows:

  min(cost) = alpha * error + beta * (num_feat / max_feat) where,

  - alpha * error: Error represents the loss function multiplied by a coefficient alpha, which represent the weight assigned to the error term. alpha = 0.99
  - beta * (num_feat / max_feat): It is a regularization component. num_feat is the number of features, while max_feat is the maximum allowable number of features. beta = (1-alpha)
- Flower Population: In FPO algorithm, each potential solution is represented as a ‘flower’. They are categorized into three types: ‘source flowers,’ ‘donor flowers’ and ‘receptor flowers’.
- Source Flowers: They represent the current best solutions in the population.
- Donor Flowers: They are the solutions with lower fitness values. They act as pollinators that carry information to receptor flowers.
- Receptor Flowers: They carry the information that they receive from the donor flowers. These flowers update their positions based on the information received from the donors. The pollination process entails an exchange of information between donor and receptor flowers to update the position of the receptor flower.
- Pollination Process: The core of the FPO is its pollination process, which presents the way pollination process is carried out by the bees. The positions of the flowers is updated iteratively based on the positions of the donor flowers and source flowers. The algorithm uses a switch probability ‘p’ to decide when to perform global or local pollination. For each solution a random number is generated, and if it is less than ‘p’, global pollination is performed; otherwise, local pollination takes place. The pollination process is affected by parameters like the step size, rate of pollination, population size and number of generations.
- Termination: The algorithm repeats the pollination process for a specified number of cycles or until a convergence criterion is met. The best solution found during this optimization is regarded as the algorithm's final output.

After initializing the global iteration, there are two possible pollinations that can happen, that is global or local pollination. Global pollination is carried out by insects such as bees that can travel long distances. Pollen grains of the flowers are carried over longer distances because insects can fly and cover a larger range. This process ensures the pollination of the fittest features. Local pollination occurs within limited range when natural elements like rain or wind carry pollen grains to nearby areas. Local pollination occurs within a limited range. Among all the global and local features, the best features are selected based on the threshold value and the fitness value. Following are some of the best features selected using FPO algorithm for the functional and non-functional requirements classification :
‘claim', 'available', 'authorized', 'consistently', 'status', 'generated', 'maintained', 'security', 'replenishment', 'responsibility', 'royalty', 'satisfaction', 'replacement', 'service', 'solution', 'damage', etc.

Through iterations, the FPO algorithm converges on a feature subset that optimally balances the trade-off between classification performance and feature dimensionality. The size, number of selected features and the error rate, are calculated to determine the best FPO scores. Based on these scores, the most relevant features are chosen.

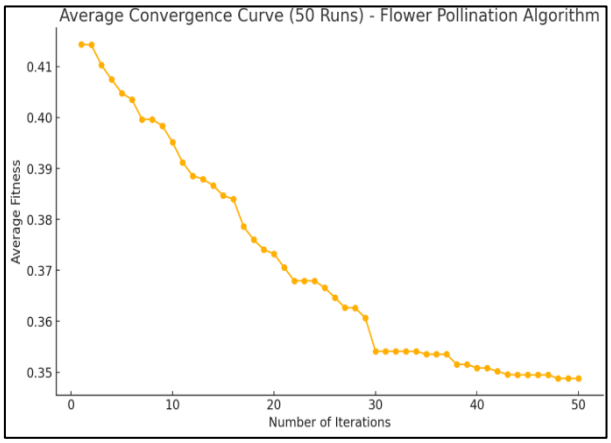


Figure 14: Performance of the FPO algorithm using Word2Vec for 50 runs

The above Figure 14. illustrates the average fitness scores for the FPO algorithm across 50 runs. It provides insights into the optimization behavior and performance of the algorithm across multiple iterations. The detailed analysis is as follows:

- Exploratory Phase: At the beginning of the curve for the first 10-15 iterations, there is a noticeable and rapid decrease in the average fitness value. This indicates that the algorithm is effectively exploring the search space and finding better solutions quickly. The global pollination phase, driven by Lévy flights, is likely helping the algorithm make significant strides in fitness improvement during these early stages.
- Transition Phase: Between iterations 15 to 35, the average fitness continues to decrease, but the fitness curve is less steep than in the initial phase. This phase indicates that the algorithm is moving gradually into a more refined search, balancing global exploration with local exploitation.
- Final Convergence phase: After around iteration 35, the curve begins to flatten, and fitness values stabilize with only minimal decreases in each subsequent iteration. This flattening of the curve suggests that the solutions are nearing optimality, and additional iterations bring only slight improvements, if any.
- Overall Convergence Behavior: The graph shows a typical convergence pattern with a rapid decrease in fitness initially, followed by a gradual tapering off. The algorithm demonstrates effective convergence behavior, indicating robustness in handling the optimization problem.

The overall trend of the graph demonstrates how FPO is effective in global optimization tasks, progressively minimizing the fitness value to find better solutions until it converges on near-optimal feature set for the classification task. The following Figure 15. represents the flowchart of the propsed approach.

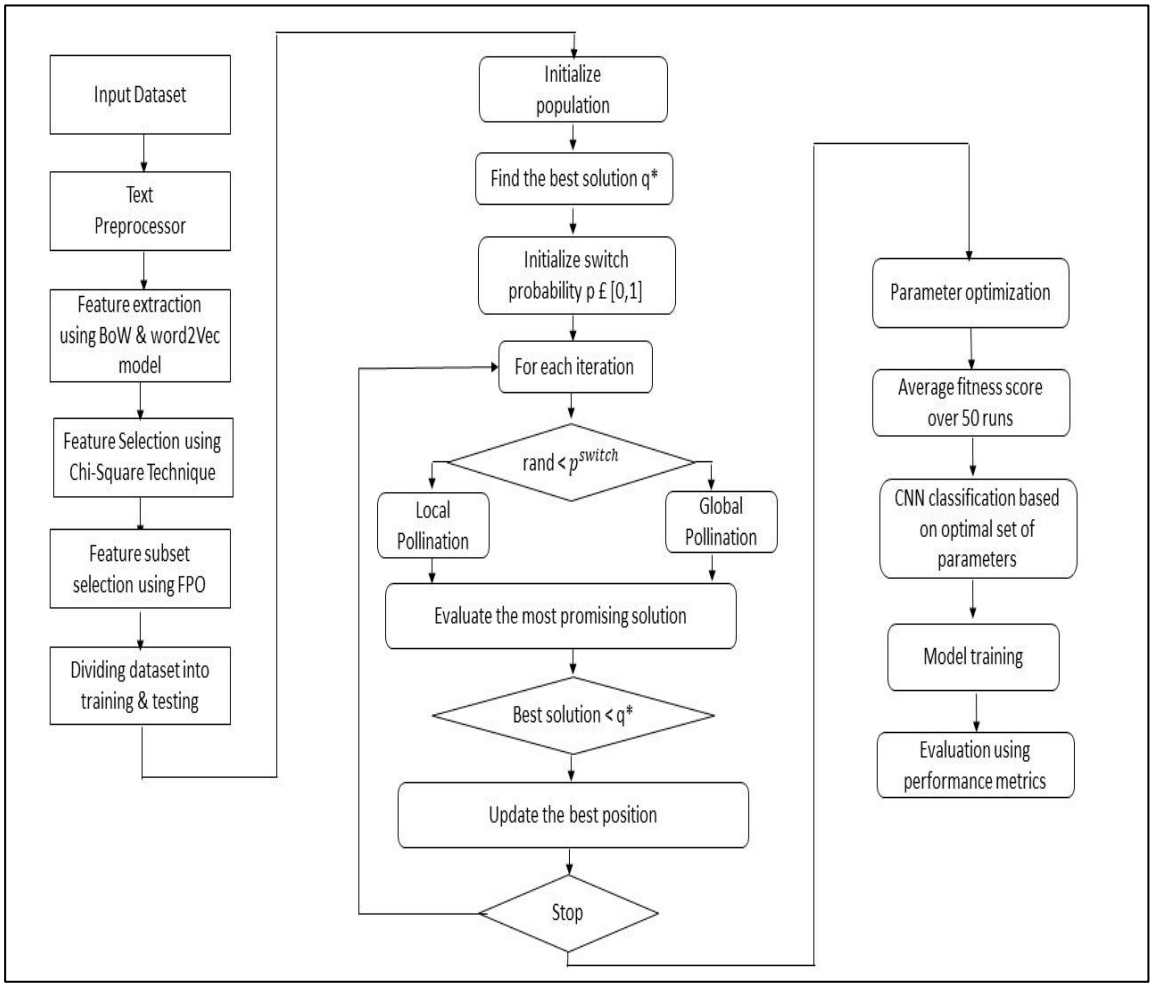


Figure 15: CNN-FPO approach

The pseudocode of the FPO algorithm is given as follows:

FPO ALGORITHM: Pseudocode

Objective: Minimize or maximize f(y), where x = (y1, y2, ..., yd)

1. Initialize a population of m pollen gametes/ flowers with random solutions
2. Find the current best solution q* in the initial population
3. Define a switch probability p ∈ [0, 1]
4. while (Z < max_no_generation)
5. for i = 1 to m (for all m flowers in the population)
6. if rand < p,
7. Draw a step vector L of dimensional d which obeys a Lévy distribution
8. Global pollination via $y_i^{(z+1)} = y_i^z + L(q^* - y_i^z)$
9. else
10. Draw ϱ from a uniform distribution in [0,1]
11. Randomly choose a and b among all the solutions
12. Do local pollination via $y_i^{(z+1)} = y_i^z + \varepsilon(y_a^z - y_b^z)$
13. end if
14. Evaluate new solutions
15. If new solutions are better, update them in the population
16. end for
17. Find the current best solution q*
18. end while

The pseudocode above describes the FPO algorithm, which balances global exploration using Lévy flight steps with local exploitation. Key parameters of the FPO algorithm include switch probability (p), population size (m) and maximum number of generations (t), all of which influence the algorithm's efficiency and convergence. FPO algorithm optimizes parameters based on the data that trains the model to prevent data leakage from the test set, helping the model generalize better to new data. Table 4 lists the control parameters for FPO. To validate the results, accuracy is measured using the KNN classifier.

Table 4: Performance of FPO using BoW and Word2vec vectorizer

	Using BoW	Using Word2Vec
No. of features selected	20	205
Maximum number of generations	50	50
Switch probability	0.3	0.8
Population size	10	5
K-value in KNN	10	2
Accuracy of FPO using KNN	96.47%	81.83%

The computational requirements of the FPO algorithm combined with KNN for feature selection can be analyzed by examining the operations performed during each iteration. The complexity analysis indicates that FPO algorithm scales linearly with the number of generations (max_no_generation), population size (m), and dimensionality of the solution space (d). However, as the number of generations increase, the algorithm's runtime can increase considerably, necessitating careful parameter tuning. The computational demand also impact processing power, memory usage and the execution time. Large values of n, Max_Generation, and d increase memory, CPU, and time demands.

By integrating chi-square feature selection with the FPO algorithm, the strengths of both the methods are exploited to achieve a more refined and optimized subset of features for classification tasks.

4 Experimental evaluation results and discussion

4.1 Performance measuring parameters

The performance of the CNN-FPO framework is evaluated using various metrics, including accuracy, precision, recall, and F1 score. In this research, accuracy is assessed using classification elements like: True Positives (TP), which are correctly identified positive instances; True Negatives (TN), which are correctly identified negative instances; False Negatives (FN), which are positive instances incorrectly classified as negative; and False Positives (FP), which are negative instances incorrectly classified as positive. These elements are used to generate a confusion matrix, which aids in solving classification problems where the output can belong to multiple classes. Mathematically, these metrics are determined as follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$F1\ score = \frac{2 * Precision * Recall}{Precision+Recall} \quad (4)$$

Following Figure 16. represents the confusion matrix of the proposed CNN-FPO classifier using BoW technique. Confusion matrix shows data in the form of requirement words.

Confusion matrix reveals a strong performance by the classifier. As observed, the number of (TP) is 235, (TN) is 17, (FP) is 3 and (FN) is 0. With 235 (TP), the model correctly identified 235 instances of the positive class. The 17 (TN) indicate that the model correctly identified 17 instances of the negative class. The 3 (FP) suggest that only a small number of negative instances were incorrectly classified as positive, while the 0 (FN) highlight that the model did not miss any positive instances, indicating excellent recall. Overall, this indicates that the classifier is performing well, with minimal errors in predicting the positive class and a perfect record in identifying all positive instances.

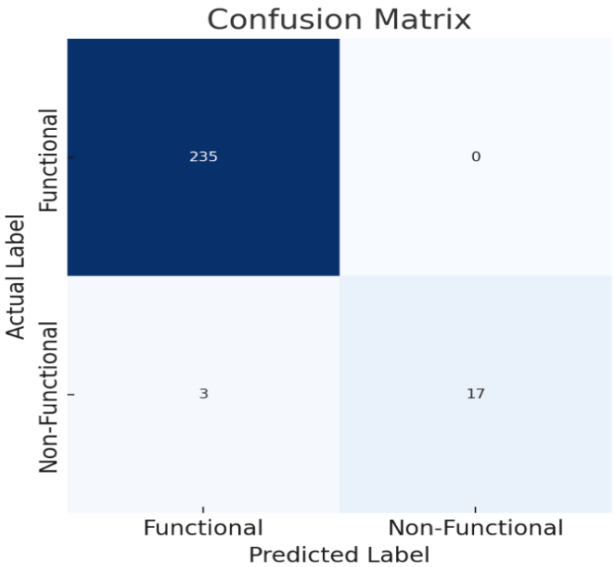


Figure 16: Confusion matrix of the proposed CNN-FPO framework using BoW model

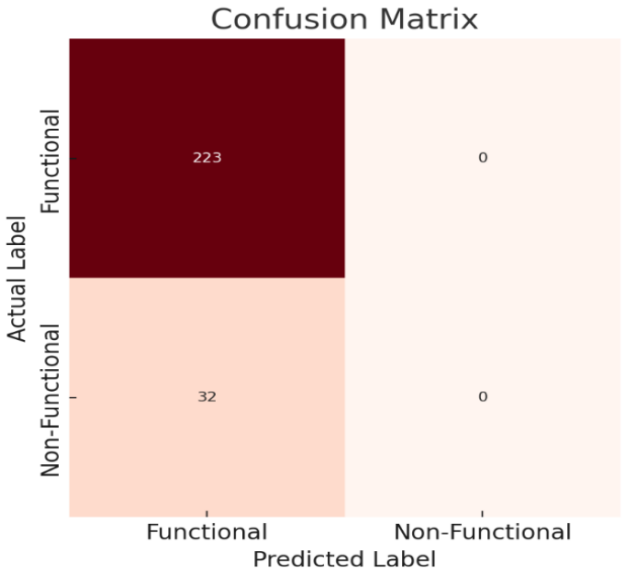


Figure 17: Confusion matrix of the proposed CNN-FPO framework using Word2Vec model

The outcome of the CNN-FPO framework using Word2Vec technique is as shown in Figure 17. The confusion matrix presents a mixed performance by the classifier. With 223 (TP), the model correctly identified a significant number of positive instances. However, the 0 (TN) and 32 (FP) suggest that the model incorrectly classified all negative instances as positive. The 0 (FN) show that the model successfully identified all positive instances, which is a positive aspect in terms of recall. While the model is effective at identifying positive instances, its inability to correctly classify any negative instances highlights a substantial weakness.

4.2 Results and discussion

In this section, we present the experimental results and discuss their implications. SVM, KNN, and CNN models were implemented using two feature representation model: Bag-of-Words (BoW) and Word2Vec. Additionally, the proposed CNN-FPO framework was applied to both BoW and Word2Vec to enhance accuracy. To assess the impact of NLP techniques on classification outcomes, the classifier parameters were held constant as follows:

- K-Nearest Neighbour
  For KNN classifier, we used scikit-learn with KNeighborsClassifier class. The parameters for this classifier were set as : [n_neighbors=3, p: 2]. p=2 corresponds to the Euclidean distance.
- SVM
  For SVM classifier, scikit-learn with SVM package was used. The parameters used in the experiments were: [kernel='poly' , degree=10, max_iter=100]
- CNN
  For all CNN based experiments, keras tensorflow library is used. It is an open-source library written in python. For the proposed

CNN-FPO framework, the input dimension to CNN was fixed to 205 using Word2Vec model and 20 using BoW model which is equal to vector dimension. The number of output neurons is equal to 2, that is same as the number of requirement categories. The rest of the parameters were fixed as: [Conv layer = 2, pooling layer = 2, No. of Kernals = [512,128,64], kernel_size = [3,1], Target classes = 2, batch size = [64,2], epochs = [3,5], optimizer = 'adam', activation output= sigmoid, loss = 'binary_crossentropy'].

Following Table 5 and Table 6 shows the comparative analysis of the classifiers using Word2Vec and BoW model for SmartNet dataset respectively. The experiments are implemented using a (70%-30%) dataset split, with 70% allocated for training and remaining 30% reserved for testing.

Table 5:Comparative analysis of the classifiers with the proposed CNN-FPO using Word2Vec model

	Accuracy	Precision	Recall	F1-score
CNN	93.16%	93.25%	100%	96.91%
SVM	80.56%	90.36%	96.59%	92.14%
KNN	91.92%	87.48%	100%	91.75%
CNN-FPO	88.45%	88.51%	100%	93.81%

Table 6: Comparative analysis of the classifiers with the proposed CNN-FPO using BoW model

	Accuracy	Precision	Recall	F1-score
CNN	97.19%	97.54%	100%	98.73%
SVM	100%	99.21%	100%	100%
KNN	97.21%	97.11%	100%	98.48%
CNN-FPO	97.89%	97.9%	100%	98.69%

The performance of various classifiers with the proposed CNN-FPO was evaluated using both Word2Vec and BoW representations focusing on accuracy, precision, recall, and F1-score. The detailed evaluation of the result is presented as follows-

1. For the Word2Vec model as shown in Table 5, the proposed CNN-FPO framework achieved a lower accuracy of 88.45%, precision of 88.51%, recall of 100%, and F1-score of 93.81%. While the CNN model achieved an accuracy of 93.16%, precision of 93.25%, recall of 100%, and F1-score of 96.91%. KNN showed similar performance to CNN with an accuracy of 91.92% and a perfect recall score. The SVM model has the lowest accuracy of 80.56%, precision of 90.36%, recall of 96.59%, and F1-score of 92.14%. With Word2Vec model, CNN-FPO, CNN and KNN achieved perfect recall scores, meaning they correctly identified all positive instances in the dataset. However, CNN-FPO and KNN has the lowest precision scores of 88.51% and 87.48% respectively. This indicates its inability to correctly classify negative instances, as discussed in Section 4.1 for CNN–FPO using Word2Vec.
2. For the BoW model, as shown in Table 6, the proposed CNN-FPO framework demonstrated excellant performance, surpassing the original CNN and other ML models with an accuracy of 97.89%, a precision of 97.9%, a recall of 100%, and an F1-score of 98.69%. Both CNN and KNN models achieved an accuracy of around 97%, with the CNN recording a precision of 97.54%, a recall of 100%, and an F1-score of 98.73%, while KNN had a precision of 97.11%, a recall of 100%, and an F1-score of 98.48%. The SVM model attained perfect scores, achieving 100% accuracy, recall, and F1-score, with a precision of 99.21%. This overfitting, however, is likely due to the small dataset, affecting the validation results. The straightforward nature of the BoW model, which simply counts word frequencies, does not mitigate the impact of the dataset's imbalanced nature

To summarize, the higher accuracy with BoW model could be due to its simplicity and focus on the most frequent features, especially when the model's task is heavily influenced by word frequency rather than context. Meanwhile, Word2Vec's nuanced representation might not be as effective on the data, particularly when the classes are not fairly distributed making the model difficult to generalize.

In this research, the CNN model is trained using training samples with a specified batch size and number of iterations. The batch size indicates the number of samples the model processes at every iteration, while epochs denote the number of times the model runs through the entire training dataset. After training, the accuracy and loss metrics are evaluated and plotted against the number of epochs, as illustrated in the figures below. For experimentation, the number of

epochs is set to 4. Blue line represents training data, while the orange line represents validation data.

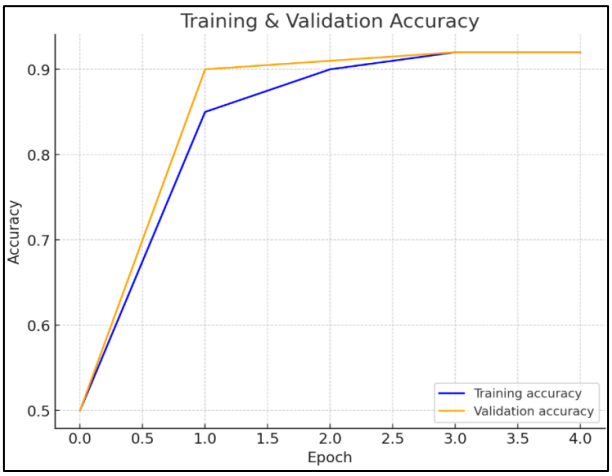


Figure 18: CNN model training and validation accuracy

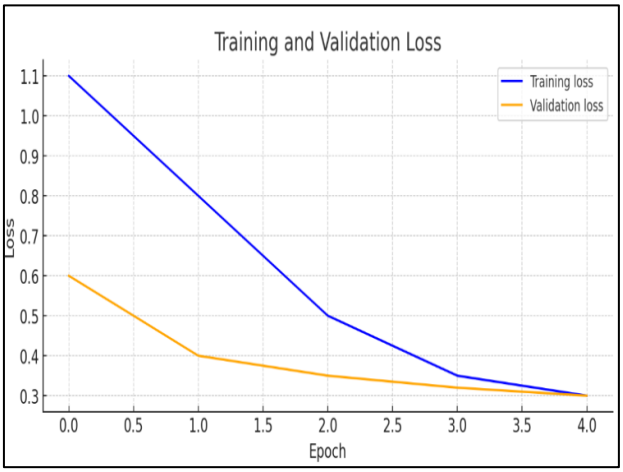


Figure 19: CNN model training and validation loss

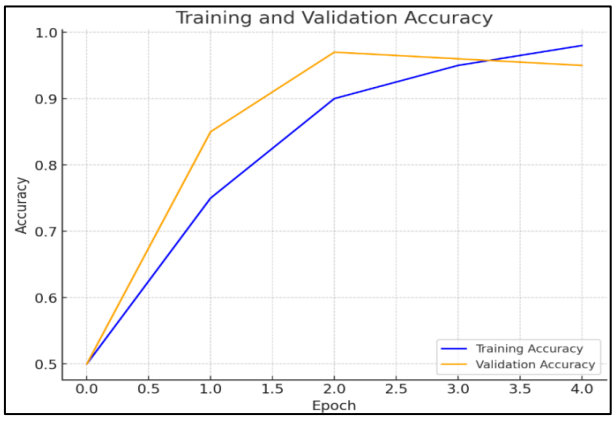


Figure 20: CNN-FPO model for training and validation accuracy

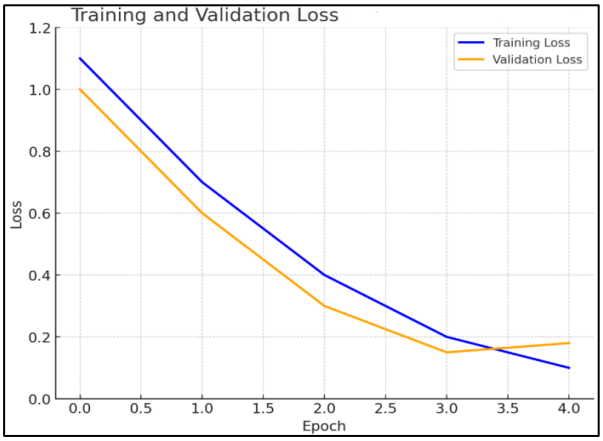


Figure 21: CNN-FPO model for training and validation loss

Figure 18. and Figure 19. illustrate the accuracy and loss of the CNN model, while Figure 20. and Figure 21. show the accuracy and loss of the proposed CNN-FPO framework. For CNN, validation accuracy stabilizes with the training accuracy early at around 0.92, indicating strong and consistent generalization throughout training. In contrast, CNN-FPO initially achieves a higher accuracy, reaching upto 0.98. However around epoch 2, the validation accuracy declines to around 0.95, suggesting a possible reduction in generalization. While Furthermore, Figure 19. and Figure 21. reveal a decrease in validation loss for the proposed CNN-FPO as compared to CNN alone. CNN maintains a stable validation loss which decreases over the epochs, indicating that the model is learning effectively and improving its performance on the validation dataset. The gradual decline in validation loss, along with its convergence towards the training loss, suggests that the model is not overfitting and is generalizing well to unseen data. Likewise, validation loss for CNN-FPO decreases, reaching approximately 0.18 by epoch 4.0. However, unlike CNN, CNN-FPO experiences a slight increase in validation loss after 3 epoch, indicating a bend towards overfitting. The CNN-FPO framework performs exceptionally well, making it ideal for tasks that require prolonged training. However, after a certain point, the validation accuracy begins to decrease, and the validation loss starts to increase, indicating the onset of overfitting.

While the model has achieved success, the thorough examination of the results are discussed below-

1. The Bag of Words (BoW) feature extraction technique identified 372 unique words, while the Word2Vec technique extracted 401 unique words. For example, BoW omitted words like 'first', 'one', 'six', 'third', and 'two' due to its default tokenization rules, which often exclude words containing numbers. Additionally, BoW discards words such as 'hence', 'get', 'give', and 'go', treating them similarly to stopwords.
2. The proposed CNN-FPO framework demonstrates robust performances for both BoW

and Word2Vec model, especially excelling with BoW due to its simpler representation. Word2Vec fell short because its strength lies in generating meaningful word embeddings that excel when all classes are fairly represented.

3. ML algorithms like SVM and KNN performed well with the BoW feature extraction technique, whereas CNN performed better when used with Word2Vec. This is due to BoW's compatibility with SVM and KNN, which benefit from its straightforward and sparse representation, whereas Word2Vec offers denser representations to capture semantic relationships in data more effectively, which the complex architecture of CNNs can better leverage.
4. The findings obtained might be limited to the specific dataset used in this study. Any changes made in the requirements could impact the performance of the CNN-FPO model. Likewise, as discussed in Section 4.2, model is likely to overfit due to small dataset. Also use of data balancing technique can improve the performance of Word2Vec model.

The graphical representation of the performances of the classifiers, for BoW and Word2Vec feature extraction techniques, in terms of evaluation metrics is depicted in Figure 22. and Figure 23. respectively.

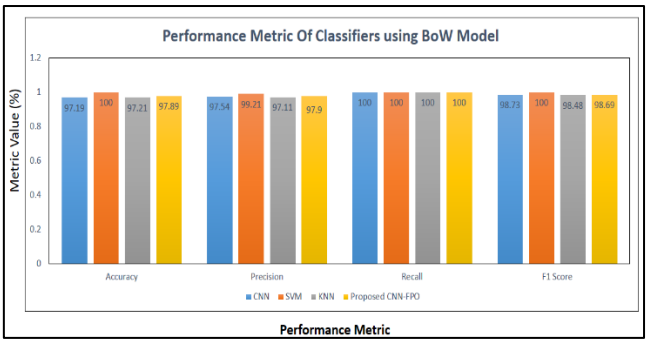


Figure 22: Comparison of the proposed and existing classification models using BoW model

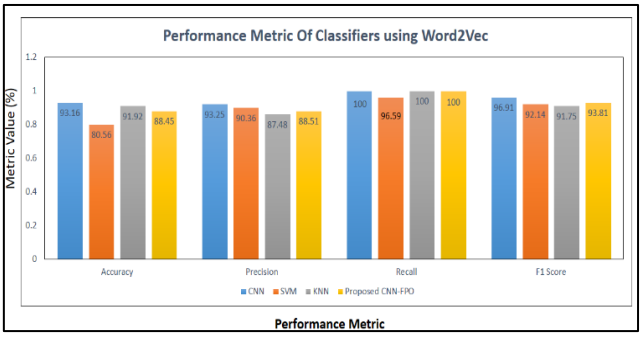


Figure 23: Comparison of the proposed and existing classification models using Word2Vec

Upon comparing the performance metrics of the state-of-the-art classifiers with the proposed CNN-FPO for both Bow and Word2Vec model, it is evident that the classifiers exhibit different levels of effectiveness under varying data conditions.

1. The BoW model shows higher accuracy across all classifiers as compared to the Word2Vec model. The proposed CNN-FPO framework shows higher accuracy with the BoW model (97.89%) than with Word2Vec (88.45%).
2. Precision is generally higher with the BoW model compared to the Word2Vec model. The difference is particularly stark for KNN and CNN-FPO, where precision drops more noticeably with Word2Vec.
3. Recall is consistently perfect across most of the classifiers for both the models.
4. F1-score is again higher across all classifiers when using the BoW model compared to the Word2Vec model. CNN-FPO particularly shows a drop in F1-score when using Word2Vec.
5. Overall, BoW model appears to be better suited for the SmartNet dataset, leading to superior performance for all the classifiers as compared to the Word2Vec model. Word2Vec model shows lower performance across the board, which may be attributed to its sensitivity to dataset balance.

5 Threats to validity

When classifying requirements using the industry SmartNet dataset, it is crucial to consider potential threats to both external and internal validity.

5.1 Internal threats to validity

- Measurement Error: Inaccurate requirement labeling in the dataset could result in flawed model training and validation. To minimize this risk, a team of experts performs data labeling according to industry standards and procedures, ensuring both accuracy and consistency.

5.2 External threats to validity

- Temporal Validity: An outdated dataset can lead to poor model performance on current data due to evolving technology and standards. To counter this, the dataset used in this research is sourced from real-world operations and projects within the SmartNet system, ensuring it reflects up-to-date conditions. This data captures real-time requirements encountered during the system's operation and customer service delivery, making it highly relevant and valuable for training and validating models in real-world contexts.

- Construct validity- The concepts of FR and NFR can vary in interpretation across industries, impacting how results are understood and utilized. To address this, FR and NFR are clearly defined in this research, following definitions provided by experts at Godrej.

Mitigating these internal and external validity threats is essential to ensure that the findings from the classification study are robust and reliable.

6 Conclusion and future scope

This paper examined the performance of the CNN model for classifying requirement specification into functional and non-functional requirements using data preprocessing tools, such as NLP. The essential features are extracted using two feature extraction techniques namely Word2Vec and BoW and the significant features are selected using a Chi-square method. Further the performance of the model is improved by optimizing CNN using FPO. The proposed CNN-FPO framework selects the most relevant features to enhance the performance of the classifier. By selecting the subset of relevant features, the computational performance of the CNN model is raised. The effectiveness of the proposed approach is validated from simulation results where it is observed that, the training loss is lesser than the validation loss and the validation accuracy is lower than the training accuracy. The performance of the proposed CNN-FPO framework is assessed using an industry SmartNet dataset. In addition, the performance of the proposed CNN-FPO approach is compared with other benchmark models such as SVM, KNN and CNN. It is seen that the accuracy of the proposed CNN-FPO framework reaches upto 98% as compared to other classifiers. Further, the proposed CNN-FPO framework works best with BoW model as compared to Word2Vec. Even with this performance, the obtained findings might be restricted only to the specific dataset used in this work. In addition, the CNN-FPO model might not consider the interdependencies between the requirements, which could affect the accuracy.

Future work involves enhancing the generalizability of the findings, by validating the model on datasets from various domains like finance, telecommunications healthcare etc. Additionally future work also intends to mitigate the effect of data imbalance by using data balancing techniques and examine the impact of data representation methods.

Acknowledgement

We would like to express our deepest gratitude to Dr.Dhirendra Mishra, HoD of Computer Engineering department, SVKM's NMIMS Mukesh Patel School of Technology Management & Engineering (MPSTME), for his incredible support and encouragement throughout the course of this research. We also acknowledge that this research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] Van Lamsweerde (2009). Requirements engineering: From system goals to UML models to software, Chichester, *UK: John Wiley , Sons.* vol. 10.

[2] Fantechi A, Gnesi S, Semini L. (2023). VIBE: looking for variability in ambiguous requirements. *Journal of Systems and Software*, 1;vol.195:111540. https://doi.org/10.1016/j.jss.2022.111540.

[3] Dabbagh, M., & Lee, S. P. (2014). An approach for integrating the prioritization of functional and nonfunctional requirements. *The Scientific World Journal*, 2014(1), 737626. https://doi.org/10.1155/2014/737626

[4] Dabbagh, M., Lee, S. P., & Parizi, R. M.(2016). Functional and non-functional requirements prioritization: empirical evaluation of IPA, AHP-based, and HAM-based approaches. *Soft Computing*, 20, 4497-4520. https://doi.org/10.1007/s00500-015-1760-z

[5] Gruber, K., Huemer, J., Zimmermann, A., & Maschotta, R. (2017). Integrated description of functional and non-functional requirements for automotive systems design using SysML. *In: 7th IEEE International Conference on System Engineering and Technology (ICSET)*, Shah Alam, Malaysia, pp. 27-31. DOI: 10.1109/ICSEngT.2017.8123415

[6] Frattini, J., Montgomery, L., Fischbach, J., Mendez, D., Fucci, D., & Unterkalmsteiner, M. (2023). Requirements quality research: a harmonized theory, evaluation, and roadmap. *Requirements Engineering*, 28(4), 507-520. https://doi.org/10.48550/arXiv.2309.10355.

[7] Supakkul S, Chung L. (2005). Integrating FRs and NFRs: A use case and goal driven approach. *Framework*,6,p.7. https://www.researchgate.net/publication/2869405_Integrating_FRs_and_NFRs_A_Use_Case_and_Goal_Driven_Approach.

[8] Werner C, Li ZS, Lowlind D, Elazhary O, Ernst N, Damian D. (2021). Continuously managing nfrs: Opportunities and challenges in practice. *IEEE Transactions on Software Engineering*. 17;48(7):2629-42. DOI: 10.1109/TSE.2021.3066330.

[9] Mahmoud A, Williams G. (2016). Detecting, classifying, and tracing non-functional software requirements. *Requirements Engineering*. 21:357-81. https://doi.org/10.1007/s00766-016-0252-8.

[10] Rahimi, N., Eassa, F., & Elrefaei, L. (2020). An ensemble machine learning technique for functional requirement classification. *Symmetry*, 12(10), 1601. https://doi.org/10.3390/sym12101601.

[11] Shreda, Q. A., & Hanani, A. A.(2021). Identifying non-functional requirements from unconstrained documents using natural language processing and machine learning approaches. *IEEE Access*, pp.1-1 https://doi.org/10.1109/ACCESS.2021.3052921

[12] Yang H, De Roeck A, Gervasi V, Willis A, Nuseibeh B.(2011). Analysing anaphoric ambiguity in natural language requirements. *Requirements engineering*. 16:163-89. DOI:10.1007/s00766-011-0119-y.

[13] Diamantopoulos T, Roth M, Symeonidis A, Klein E. (2017). Software requirements as an application domain for natural language processing. *Language Resources and Evaluation*. 51:495-524. DOI:10.1007/s10579-017-9381-z.

[14] Dunnmon JA, Yi D, Langlotz CP, Ré C, Rubin DL, Lungren MP. (2019). Assessment of convolutional neural networks for automated classification of chest radiographs. *Radiology*. 290(2):537-44. https://doi.org/10.1148/radiol.2018181422.

[15] Watson C, Cooper N, Palacio DN, Moran K, Poshyvanyk D. (2022). A systematic literature review on the use of deep learning in software engineering research. *ACM Transactions on Software Engineering and Methodology (TOSEM)*. 4;31(2):1-58. https://doi.org/10.1145/3485275

[16] Sagar VB, Abirami S. (2014). Conceptual modeling of natural language functional requirements. *Journal of Systems and Software*. 1;88:25-41. https://doi.org/10.1016/j.jss.2013.08.036

[17] McGraw, K. L., & Harbison, K. (2020). User-centered requirements: The scenario-based engineering process. *CRC Press*.

[18] Jeffrey HJ, Putman AO. (1994). Relationship definition and management: tools for requirements analysis. *Journal of Systems and Software*. 1;24(3):277-94. https://doi.org/10.1016/0164-1212(94)90069-8.

[19] Surma-aho, A., Björklund, T., & Hölttä-Otto, K. (2022). User and stakeholder perspective taking in novice design teams. *Design Science*, 8, e24. https://doi.org/10.1017/dsj.2022.19

[20] Fernández, D. M., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., ... & Wieringa, R. (2017). Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. *Empirical Software Engineering*, 22, 2298-2338. https://doi.org/10.1007/s10664-016-9451-7

[21] Blake R, Mangiameli P. (2011). The effects and interactions of data quality and problem complexity on classification. *Journal of Data and Information Quality (JDIQ)*. 1;2(2):1-28. https://doi.org/10.1145/1891879.1891881

[22] Nguyen TH, Grundy J, Almorsy M. (2015). Rule-based extraction of goal-use case models from text. *In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering.* pp. 591-601. https://doi.org/10.1145/2786805.2786876

[23] Vlas R, Robinson WN. (2011). A rule-based natural language technique for requirements discovery and classification in open-source software development projects. *In 2011 44th Hawaii International Conference on System Sciences.* pp. 1-10. IEEE. DOI:10.1109/HICSS.2011.28

[24] Singh, P., Singh, D., & Sharma, A. (2016). Rule-based system for automated classification of non-functional requirements from requirement specifications. *In: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 620-626. https://doi.org/10.1109/ICACCI.2016.7732115

[25] Sharma, V. S., Ramnani, R. R., & Sengupta, S. (2014). A framework for identifying and analyzing non-functional requirements from text. *In: Proceedings of the 4th International Workshop on Twin Peaks of Requirements and Architecture*, pp. 1-8. https://doi.org/10.1145/2593861.2593862

[26] Hussain, I., Kosseim, L., & Ormandjieva, O. (2008). Using linguistic knowledge to classify non-functional requirements in SRS documents. *In: Natural Language and Information Systems: 13th International Conference on Applications of Natural Language to Information Systems*, NLDB 2008 London, UK, June 24-27, Springer Berlin Heidelberg, pp. 287-298. https://doi.org/10.1007/978-3-540-69858-6_28

[27] Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (2007). Automated classification of non-functional requirements. *Requirements Engineering*, 12, 103-120. https://doi.org/10.1145/3084226.3084241.

[28] Samantaray SR. (2013). A systematic fuzzy rule based approach for fault classification in transmission lines. *Applied soft computing*, Feb 1;13(2):928-38. https://doi.org/10.1016/j.asoc.2012.09.010

[29] Das S, Deb N, Cortesi A, Chaki N. (2024). Extracting goal models from natural language requirement specifications. *Journal of Systems and Software*, 211:111981. https://doi.org/10.1016/j.jss.2024.111981

[30] Casamayor A, Godoy D, Campo M. (2010). Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Information and Software Technology*, 52(4):436-45. https://doi.org/10.1016/j.infsof.2009.10.010

[31] Pitangueira AM, Maciel RS, Barros M. (2015). Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature. *Journal of Systems and Software*, 1;103:267-80. https://doi.org/10.1016/j.jss.2014.09.038

[32] Abad ZS, Karras O, Ghazi P, Glinz M, Ruhe G, Schneider K. (2017). What works better? a study of classifying requirements. *In 2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 496-501. IEEE. DOI: 10.1109/RE.2017.36

[33] Li, L. F., Jin-An, N. C., Kasirun, Z. M., & Chua, Y. P. (2019). An empirical comparison of machine learning algorithms for classification of software requirements. *International Journal of Advanced Computer Science and Applications*, 10(11), 258-263. https://doi.org/10.14569/IJACSA.2019.0101135

[34] Handa, N., Sharma, A., & Gupta, A. (2022) Framework for prediction and classification of non-functional requirements: A novel vision. *Cluster Computing*, 25(2), 1155-1173. https://doi.org/10.1007/s10586-021-03484-0

[35] Khan A, Baharudin B, Lee LH, Khan K. (2010). A review of machine learning algorithms for text-documents classification. *Journal of advances in information technology*, 1(1):4-20. DOI:10.4304/jait.1.1.4-20

[36] Quba, G. Y., Al Qaisi, H., Althunibat, A., & AlZu'bi, S. (2021). Software requirements classification using machine learning algorithms. *In: 2021 International Conference on Information Technology (ICIT)*, Amman, Jordan, pp. 685-690. DOI:10.1109/ICIT52682.2021.9491688

[37] Jindal R, Malhotra R, Jain A. (2016). Automated classification of security requirements. *In 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2027-2033. IEEE. DOI: 10.1109/ICACCI.2016.7732349

[38] Dave, Dev Jayant. (2022). Identifying Functional and Non-functional Software Requirements from User App Reviews and Requirements Artifacts. Thesis, *Montclair State University*. https://digitalcommons.montclair.edu/cgi/viewcontent.cgi?article=2014&context=etd.

[39] Raymond, R., & Savarimuthu, M. A. (2023). Retrieval of Interactive requirements for Data Intensive Applications using Random Forest Classifier. *Informatica*, 47(9). https://doi.org/10.31449/inf.47i9.3772

[40] Binkhonain, M., & Zhao, L. (2019). A review of machine learning algorithms for identification and classification of non-functional requirements. *Expert Systems with Applications*, 150, 1139-1152. https://doi.org/10.1016/j.eswa.2020.113990

[41] Kurtanović, Z., & Maalej, W. (2017). Automatically classifying functional and non-functional requirements using supervised machine learning. *In: 2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 490-495. https://doi.org/10.1109/RE.2017.82

[42] Slankas, J., & Williams, L. (2013) Automated extraction of non-functional requirements in available documentation. *In: 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE)*, San Francisco, CA, USA, pp. 9-16. DOI:10.1109/NAturaLiSE.2013.6611715

[43] AlOmar EA, Mkaouer MW, Ouni A. (2021). Toward the automatic classification of self-affirmed refactoring. *Journal of Systems and Software*. 171:110821. https://doi.org/10.1016/j.jss.2020.110821.

[44] Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E. V., & Batista-Navarro, R. T. (2021). Natural language processing for requirements engineering: A systematic mapping study. *ACM Computing Surveys (CSUR)*, 54(3), 1-41. https://doi.org/10.1145/3444689

[45] Amasaki, S., & Leelaprute, P. (2018). The effects of vectorization methods on non-functional requirements classification. *In: 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 175-182. https://doi.org/10.1109/SEAA.2018.00036

[46] Tiun, S., Mokhtar, U. A., Bakar, S. H., & Saad, S. (2020). Classification of functional and non-

functional requirement in software requirement using Word2Vec and FastText. *Journal of Physics: Conference Series*, 1529(4), 042077. https://doi.org/10.1088/1742 6596/1529/4/042077

[47] Sabir, M., Chrysoulas, C., & Banissi, E. (2020). Multi-label classifier to deal with misclassification in non-functional requirements. *In: Trends and Innovations in Information Systems and Technologies*, *Springer International Publishing*, Volume 1, pp. 486-493. https://doi.org/10.1007/978-3-030-45688-7_49

[48] Song, D., Vold, A., Madan, K., & Schilder, F. (2022). Multi-label legal document classification: A deep learning-based approach with label-attention and domain-specific pre-training. *Information Systems*, 106, 101718. https://doi.org/10.1016/j.is.2021.101718

[49] Jiang, J. Y., Tsai, S. C., & Lee, S. J. (2012). FSKNN: multi-label text categorization based on fuzzy similarity and k nearest neighbors. *Expert Systems with Applications*, 39(3), 2813-2821. https://doi.org/10.1016/j.eswa.2011.08.141

[50] Ramadhani, D. A., Rochimah, S., & Yuhana, U. L. (2015). Classification of non-functional requirements using semantic-FSKNN based ISO/IEC 9126. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 13(4), 1456-1465. https://doi.org/10.12928/telkomnika.v13i4.2300

[51] AlDhafer, O., Ahmad, I., & Mahmood, S. (2022). An end-to-end deep learning system for requirements classification using recurrent neural networks. *Information and Software Technology*, 147, 106877. https://doi.org/10.1016/j.infsof.2022.106877

[52] Winkler, J., & Vogelsang, A. (2016). Automatic classification of requirements based on convolutional neural networks. *In: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pp. 39-45. DOI: 10.1109/REW.2016.021

[53] Baker, C., Deng, L., Chakraborty, S., & Dehlinger, J. (2019). Automatic multi-class non-functional software requirements classification using neural networks. *In: 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, pp. 610-615. https://doi.org/10.1109/ICIT52682.2021.9491688

[54] Gnanasekaran, R. K., Chakraborty, S., Dehlinger, J., & Deng, L. (2021). Using recurrent neural networks for classification of natural language-based non-functional requirements. *In: REFSQ Workshops*, Essen, Germany. Corpus ID: 235076554

[55] Li, G., Zheng, C., Li, M., & Wang, H. (2022). Automatic requirements classification based on graph attention network. *IEEE Access*, 10, 30080-30090. DOI: 10.1109/ACCESS.2022.3159238 https://doi.org/10.1109/ACCESS.2022.3159238

[56] Alhaizaey, A., & Al-Mashari, M. (2023). A framework for reviewing and improving non-functional requirements in agile-based requirements. *In: 2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*, Aveiro, Portugal, pp. 1-7. https://doi.org/10.23919/CISTI58278.2023.10211956.

[57] Shin J, Kim Y, Yoon S, Jung K. (2018). Contextual-CNN A novel architecture capturing unified meaning for sentence classification. *In 2018 IEEE international conference on big data and smart computing (BigComp)*, IEEE, pp. 491-494. DOI: 10.1109/BigComp.2018.00079.

[58] Li Q, Peng H, Li J, Xia C, Yang R, Sun L, Yu PS, He L. (2020). A survey on text classification: From shallow to deep learning. arXiv preprint arXiv:2008.00364.

[59] Javed, T. A., Shahzad, W., & Arshad, U. (2021). Hierarchical text classification of urdu news using deep neural network. arXiv preprint arXiv:2107.03141. https://doi.org/10.48550/arXiv.2107.03141

[60] Mullis J, Chen C, Morkos B, Ferguson S. (2024). Deep Neural Networks in Natural Language Processing for Classifying Requirements by Origin and Functionality: An Application of BERT in System Requirements. *Journal of Mechanical Design*. 146(4):041401. https://doi.org/10.1145/3444689.

[61] Hey T, Keim J, Koziolek A, Tichy WF. (2020). Norbert: Transfer learning for requirements classification. *In 2020 IEEE 28th international requirements engineering conference (RE)*. pp. 169-179, IEEE. DOI: 10.1109/RE48521.2020.00028.

[62] Khan, M. A., Khan, M. S., Khan, I., Ahmad, S., & Huda, S. (2023). Non Functional Requirements Identification and Classification Using Transfer Learning Model. *IEEE Access*. pp(99):1-1. https://doi.org/10.1109/ACCESS.2023.3295238.

[63] Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (2006). The detection and classification of non-functional requirements with application to early aspects. *In: 14th IEEE International Requirements*

Engineering Conference (RE'06), pp. 39-48. https://doi.org/10.1109/RE.2006.65.

[64] Rahman, K., Ghani, A., Ahmad, R., & Sajjad, S. H. (2023). Hybrid deep learning approach for nonfunctional software requirements classifications. *In: 2023 International Conference on Communication, Computing and Digital Systems (C-CODE)*, Islamabad, Pakistan, pp. 1-5. DOI: 10.1109/C-CODE58145.2023.10139907

[65] Saleem, S., Asim, M. N., Van Elst, L., & Dengel, A.(2016). FNReq-Net: A hybrid computational framework for functional and non-functional requirements classification. *Journal of King Saud University-Computer and Information Sciences*, 35(8), 101665. https://doi.org/10.1016/j.jksuci.2023.101665

[66] Ferrari, Alessio, Giorgio Oronzo Spagnolo, and Stefania Gnesi. (2017). Pure: A dataset of public requirements documents. *In 2017 IEEE 25th international requirements engineering conference (RE)*, pp. 502-505. DOI: 10.1109/RE.2017.29.

[67] Karim, S., Warnars, H.L.H.S., Gaol, F.L. Abdurachman, E. and Soewito, B. (2017). Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset. *In 2017 IEEE international conference on cybernetics and computational intelligence (CyberneticsCom*), pp. 19-23. IEEE. DOI: 10.1109/CYBERNETICSCOM.2017.8311708

[68] Sonawane, S. N., & Puthran, S. M. (2024). Classification of functional and nonfunctional requirements based on convolutional neural network with flower pollination optimizer. *Innovations in Systems and Software Engineering*, 1-25. https://doi.org/10.1007/s11334-024-00592-z

[69] Mullen, L. A., Benoit, K., Keyes, O., Selivanov, D., & Arnold, J. (2018). Fast, consistent tokenization of natural language text. *Journal of Open Source Software*, 3(23), 655. https://doi.org/10.21105/joss.00655.

[70] Yogish D, Manjunath TN, Hegadi RS. (2019). Review on natural language processing trends and techniques using NLTK. *In Recent Trends in Image Processing and Pattern Recognition: Second International Conference, RTIP2R 2018*, Solapur, India, December 21–22, Revised Selected Papers, Part III (pp. 589-606). Springer Singapore. DOI:10.1007/978-981-13-9187-3_53.

[71] Mladenic D. (2002). Automatic word lemmatization. *In Proceedings of the 5th international multi-conference information society*, IS-2002 B 2002, pp. 153-159. Corpus ID: 5486846

[72] Goodman, E. L., Zimmerman, C., & Hudson, C. (2020). Packet2vec: Utilizing word2vec for feature extraction in packet data. *In: IAPR International Conference on Machine Learning and Data Mining in Pattern Recognition*. https://doi.org/10.48550/arXiv.2004.14477.

[73] Zhang Y, Jin R, Zhou ZH. (2010). Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics*, 1:43-52. DOI:10.1007/s13042-010-0001-0.

[74] Choudhary, K., & Beniwal, R. (2021). Xplore Word Embedding Using CBOW Model and Skip-Gram Model. *In: 2021 7th International Conference on Signal Processing and Communication (ICSC)*, pp. 267-270. DOI: 10.1109/ICSC53193.2021.9673321.

[75] Bahassine, S., Madani, A., Al-Sarem, M., & Kissi, M. (2020). Feature selection using an improved Chi-square for Arabic text classification. *Journal of King Saud University-Computer and Information Sciences*, 32(2), 225-231. https://doi.org/10.1016/j.jksuci.2018.05.010.

[76] Cervantes J, Garcia-Lamont F, Rodríguez-Mazahua L, Lopez A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 30;408:189-215. https://doi.org/10.1016/j.neucom.2019.10.118.

[77] Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN model-based approach in classification. *In: On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences*, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, Springer Berlin Heidelberg, pp. 986-996. https://doi.org/10.1007/978-3-540-39964-3_62.

[78] Habib G, Qureshi S. (2022). Optimization and acceleration of convolutional neural networks: A survey. *Journal of King Saud University-Computer and Information Sciences*. 1;34(7):4244-68. DOI:10.1016/j.jksuci.2020.10.004 https://doi.org/10.1016/j.jksuci.2020.10.004

[79] Abdel-Basset, M., & Shawky, L. A. (2019). Flower pollination algorithm: a comprehensive review. *Artificial Intelligence Review*, 52, 2533-2557. https://doi.org/10.1007/s10462-018-9624-4.

[80] Esa, M. F. M., Mustaffa, N. H., Radzi, N. H. M., & Sallehuddin, R. (2022). Flower Pollination Algorithm for Convolutional Neural Network

Training in Vibration Classification. *In: Computational Intelligence in Machine Learning: Select Proceedings of ICCIML*, pp. 339-346. https://doi.org/10.1007/978-981-16-8484-5_32.