Hybrid SVM-Bidirectional Long Short-Term Memory Model for Fine-Grained Software Requirement Classification

Mahmuda Akter Metu [1], Nazneen Akhter [2,*], Sanjeda Nasrin [1], Tasnim Anzum [1], Afrina Khatun [1], and Rashed Mazumder [3]

[1] Department of Information and Communication Technology, Faculty of Science and Technology, Bangladesh University of Professionals, Mirpur, Dhaka-1216, Bangladesh
[2] Department of Computer Science and Engineering, Faculty of Science and Technology, Bangladesh University of Professionals, Mirpur, Dhaka-1216, Bangladesh
[3] Institute of Information Technology, Jahangirnagar University, Savar, Dhaka-1342, Bangladesh
Email: mmahmuda092@gmail.com (M.A.M.); nazneen.akhter@bup.edu.bd (N.A.); sanjeda0390@gmail.com (S.N.); tasnimanzum1234@gmail.com (T.A.); afrina.khatun@bup.edu.bd (A.K.), rmiit@juniv.edu (R.M.)
*Corresponding Author

***Abstract*—This study focuses on advancing the classification of software requirements, particularly within the subclasses of Non-Functional requirements. Four machine learning algorithms—Support Vector Machine (SVM), Logistic Regression (LR), K-Nearest Neighbors (KNN), and Naive Bayes (NB)—were initially implemented, with SVM exhibiting superior performance. To enhance overall accuracy, a voting classifier ensemble method was employed, resulting in significant improvement. In the realm of deep learning, a standalone Bidirectional Long Short-Term Memory (Bi-LSTM) network faced challenges in fine-tuning. To harness the strengths of both machine learning and deep learning, we proposed a hybrid model by integrating SVM with Bi-LSTM. This hybrid model surpassed all prior experiments, highlighting the synergistic potential between traditional machine learning and deep learning. Our findings showcase the effectiveness of combining SVM's discriminative power with Bi-LSTM's sequential understanding, yielding a robust classification model for software requirements. This research contributes to the advancement of requirement analysis, providing a practical solution for accurately identifying diverse requirement types within the nuanced domain of Non-Functional requirements.**

***Keywords*—Software requirements, Bidirectional Long Short-Term Memory (Bi-LSTM), Support Vector Machine (SVM)**

I. Introduction

Classifying software requirements is akin to sorting puzzle pieces—it brings order to the initial jumble, making it easier to understand, manage, and ultimately, build a successful software system. Non-Functional Requirements (NFRs) are crucial in software development, defining how a system should work rather than what it should do. Classifying these NFRs helps organize, prioritize, and ensure their effective implementation. Classifying NFRs is a pivotal practice in software development, extending beyond organizational tidiness to wield substantial influence on project outcomes. By categorizing NFRs based on their impact, such as performance, security, and usability, teams can effectively prioritize and concentrate efforts on critical aspects, ensuring the development of a system that aligns with essential needs. This classification also fosters a shared language among diverse stakeholders, including developers, users, and managers, promoting clearer communication and reducing the likelihood of misunderstandings. Moreover, grouping NFRs with similar characteristics allows for targeted development efforts, optimizing resource allocation. This approach facilitates the creation of specific and relevant test cases for each NFR category, enabling thorough testing across all facets of the system. The categorization of NFRs enhances the system's maintainability, simplifying the identification and implementation of future modifications with minimized impact. Importantly, this strategic classification minimizes ambiguity, mitigating the risks of costly rework or missed deadlines, and ensures that critical features are prioritized and delivered even within budget or time constraints. In essence, NFR classification emerges not as a mere formality but as a strategic imperative, contributing to the clarity, focus, and efficiency pivotal for building successful and user-friendly software systems.

In the domain of NFR classification, research reveals pivotal gaps necessitating further exploration. Current methods often lack context sensitivity, requiring context-aware models that consider project-specific factors like domain intricacies and development methodologies. The dynamic nature of NFRs throughout the project lifecycle calls for research into techniques for dynamic

Manuscript received December 21, 2023; revised February 27, 2024; accepted May 20, 2024; published August 7, 2024.

classification and recalibration to ensure alignment with evolving requirements. Incorporating diverse stakeholder perspectives into NFR classification schemes is crucial, ensuring a holistic understanding beyond technical viewpoints. Exploring the potential of particularly Machine Learning (ML) and natural language processing, offers a promising avenue for automated NFR categorization but requires addressing challenges such as ambiguity in requirements. Research should extend beyond classification to encompass impactful methodologies for analyzing and prioritizing NFRs, informing effective resource allocation and project planning. The seamless integration and collaboration of NFR tools across development environments demand research attention to enhance collaborative classification and management. Notably, the underutilized potential of combining Machine Learning (ML) and Deep Learning (DL) for NFR classification presents a promising frontier, offering a more robust and accurate approach to address existing challenges in software development methodologies.

The aim of this work is to discover a path to get some good results by merging deep learning with machine learning in a hybrid model for multi-class classification. The plan is to utilize Bidirectional Long Short-Term Memory (Bi-LSTM)'s capacity to comprehend groupings and Support Vector Machine (SVM)'s capacity to advise contrasts between things to further develop classification accuracy and readability. This is done with the intention of developing a smarter and more adaptable model that makes use of Bi-LSTM's capacity to comprehend intricate sequences and SVM's capacity to clearly define decision limits. The hybrid model consolidates these elements to improve it at classifying different datasets, finding the right mix among accuracy and readability that is significant in some true circumstances.

The paper is organized as follows. In Section II, related works on non-functional requirements classification is presented. The detail of the proposed model is provided in Section III. The results are elaborated upon in Section IV, followed by a discussion. Finally, Section V presents the conclusion mentioning the future scope.

II. Literature Review

Zhou *et al*. [1] proposed an attention-based Long Short-Term Memory (LSTM) network for cross-language sentiment analysis, employing multilingual bidirectional LSTMs and a hierarchical attention mechanism. Demonstrated with Chinese and English, their model outperformed existing methods. Future plans include testing with new languages, datasets, and incorporating additional emotion phrases to enhance software intelligence. The research significantly contributes to understanding emotions across languages, emphasizing its essential role.

Rahman *et al.* [2] automated Non-Functional Requirements (NFR) detection in Software Requirement Specifications (SRS), a critical aspect in software development. Leveraging advanced Recurrent Neural Network (RNN), specifically Long Short-Term Memory (LSTM), and their approach achieved superior precision, recall, and F1-Score values of 0.973, 0.967, and 0.966.

Rahman *et al.* [3] investigated algorithmic hybridization for Non-Functional Requirement (NFR) classification in software engineering, employing LSTM, Bi-LSTM, and ANN approaches. Their study, based on 1000 NFR examples, revealed that the Bi-LSTM-ANN model outperforms standalone LSTM and Bi-LSTM in precision, recall, and F1-Scores.

Khayashi *et al*. [4] explored the application of deep learning techniques for categorizing software requirements using the PURE dataset. While emphasizing the importance of enhancing the precision and simplicity of the sorting process in computer systems, the study lacks a comprehensive comparison of various techniques and does not delve into real-life challenges

In Yucklar's study [5], software requirements analysis, particularly the differentiation between functional and non-functional types, is deemed crucial for successful development. The research employs AI techniques on a novel Turkish dataset, achieving a remarkable 95% accuracy with BERTurk. Despite the study's significance in the absence of similar Turkish research, it acknowledges limitations. Further exploration is needed to assess the applicability of models, address potential dataset biases, and delve into real-world implementation challenges, emphasizing the necessity for a more comprehensive solution in the realm of software development.

Kaur and Kaur [6] explored the application of the SABDM technique in the classification of software requirements, employing deep learning to distinguish between different types. However, the approach may encounter challenges with informal texts and might be perplexed by contradictory labels in the data, potentially impeding its ability to comprehend specific information.

Rahimi *et al.* [7] proposed a novel methodology, Ensemble Deep Learning, integrating LSTM, Bi-LSTM, GRU, and CNN models to categorize Software Requirements (SRs) into functional and non-functional groups. The approach allows single or two-phase classification. Despite its effectiveness, limitations include language restriction and challenges handling complex phrase structures.

In their investigation, Ali and Saleem [8] explored how computers learn to detect diverse software requirements, examining approaches like Latent Dirichlet Allocation and Naive Bayes for distinguishing Functional and Non-Functional Requirements (FRs and NFRs). Some methods, notably Latent Dirichlet Allocation and Naive Bayes, achieved a high accuracy of 95%. However, complexities arise when attempting to discern different types of NFRs, where Word2Vec techniques struggle and exhibit reduced accuracy. This review provides insights into effective and less effective methods, assisting software professionals in choosing superior approaches for understanding and organizing software needs.

Sabir's research [9] concentrates on the crucial task of determining software requirements in the software development process. Unlike earlier methods relying on

supervised machine learning, which faced challenges like insufficient labeled data and substantial setup time, Sabir aims to overcome these issues. The research explores new deep learning methods to streamline the setup of software requirements. While heading in the right direction, challenges remain, particularly in accumulating a sufficient number of labeled instances and achieving balance across various categories.

Saratha and Mukherjee [10] addressed challenges in manual requirement classification arising from diverse terminology used by stakeholders, leading to error-prone categorization. They underscored the time-consuming nature of manual classification in large projects and advocated for more precise automatic systems. The researchers discussed existing automated strategies, highlighting issues like tool complexity, limited extraction, and subpar performance

Baskoro *et al.* [11] explored software requirement classification using PROMISE and SecReq (ePurse) datasets, comparing SVM and CNN approaches with FastText feature extraction. The RNN-LSTM technique demonstrated notable performance, achieving recall, precision, and F1-Score values of 71.5%, 71.7%, and 70%, respectively, along with an accuracy of 71.5%. The study emphasized dataset-specific classification results and recommended hyperparameter adjustments to enhance CNN performance when coupled with Fasttext.

Li *et al.* [12] introduced DBGAT, a novel method for categorizing software requirements using graph attention networks and BERT. This approach exhibits remarkable accuracy and adaptability to various scenarios, surpassing constraints of conventional techniques and the uncertain applicability of LSTMs. DBGAT thoroughly examines phrase structure and word associations, achieving precision rates of up to 91% and F1-Scores of 91%, even with previously unseen projects. While this advancement sets the stage for a more effective and precise era in software development, further investigation into dynamic graph topologies and larger datasets is essential to fully harness its capabilities.

Jang *et al.* [13] scrutinized the limitations of Long Short-Term Memory (LSTM) networks, revealing challenges in achieving optimal precision, F1-Score, accuracy, and recall. Their examination underscored the drawbacks of these metrics in commonly used Natural Language Processing (NLP) techniques, emphasizing the need for innovative models capable of efficiently handling the complexities of diverse linguistic structures and data with rich context. The primary objective was to enhance the fundamental assessment criteria in natural language processing activities, recognizing the demand for advancements in addressing the intricacies of language processing tasks.

Kaur *et al.* [14] focused on software needs classification, underscoring the significance of accurate identification in software development. The investigation shed light on the deficiencies of existing manual interpretation approaches, leading to time-consuming and imprecise classifications. To address these challenges, the researchers explored the application of the BERT model with the aim of achieving more precise and automated categorization processes. The study reflects a commitment to improving the efficiency and accuracy of software needs identification through advanced and automated techniques.

NFRNet, introduced by Li and Nong [15], represents an advancement in the classification of Non-Functional Requirements (NFR) in software development. While the paper demonstrates superior performance in Precision, Recall, and F1-Score compared to current methods, it lacks a comprehensive examination of challenges related to practical implementation and computational efficiency. Despite the potential of NFRNet's automated approach in optimizing NFR identification, additional research into its practical applicability and computational efficiency would be advantageous for a more thorough understanding of its effectiveness in real-world scenarios.

Tiun *et al.* [16] investigated word embeddings and conventional features across various classifiers to identify Functional Requirements (FR) and Non-Functional Requirements (NFR) using text classification techniques. The paper highlights FastText as the superior model, showcasing its effectiveness in brief document classification compared to deep learning classifiers and conventional methods. However, the research falls short in thoroughly examining intricate models such as BERT and could benefit from additional scrutiny regarding improvements to TFIDF to enhance the accuracy of FR and NFR classification. Further exploration of advanced models and feature enhancements may contribute to a more comprehensive understanding of text classification efficacy.

Huan [17] presented a CBM model for text categorization that includes Multiscale Convolutional Neural Network (MCNN) and Bi-LSTM, overcoming the limitations of CNN and RNN. The MCNN module collected shallow local semantic information flexibly, and a MIX attention method improved important feature extraction. Experimental findings showed higher classification accuracy than benchmark datasets. Limitations included a greater network size and a longer training time. The authors' CBM model helps to advance text categorization.

In a study by Rahman *et al.*'s [18], they delved into the challenges associated with Non-Functional Requirements (NFR) using machine learning techniques. They put forward two models, DReqANN and DReqBiLSTM highlighting the drawbacks of existing methods that rely on feature extraction. The research findings showcase the performance of DReqANN in classifying NFR achieving precision ranging from 81% to an impressive 99.8% along with recall rates, between 74% and 89%. The authors emphasised the efficiency of their deep learning approach. Propose addressing complexities through transformer-based models leveraging unsupervised software requirement corpora and incorporating transfer learning methodologies.

Yahya *et al.* [19] explored an explored aspect of app development. Identifying Non-Functional Requirements (NFR) using user reviews. They introduce a learning

model that combines RNN and LSTM architectures emphasizing the lack of research on Arabic language datasets. Their model achieved a F1-Score of 96% surpassing both ML classifiers and LSTM models. The research also acknowledged limitations, such as the need for data applications and suggested potential strategies to enhance future studies through feature augmentation. Overall, this study contributes to the field by introducing an approach and highlighting the importance of considering NFR concerns, in mobile app development.

Zheng's work [20] investigated English translation text classification issues in multimedia and introduces the BATCL transfer learning technique. The study included the BERT, Att-BiLSTM, and TLCM models, demonstrating BATCL's advantages over previous techniques. It stressed transfer learning's potential to improve accuracy and efficiency in multimedia-based English translation text categorization. While precise limits, performance measures, and methods are not explicitly specified, the study makes an important contribution by presenting a unique categorization approach and proving transfer learning's efficacy in dealing with various translated texts. Tian *et al.* [21] introduced SCC-BiLSTM, a smart contract categorization model that addresses issues such as source code, comments, and account data. The model outperforms other classification models by utilizing Bi-LSTM, Gaussian LDA, and attention processes. While precise performance measures and constraints are not clearly stated, this effort helps to further smart contract research.

Huan [22] presented a CBM model for text categorization that includes Multiscale Convolutional Neural Network (MCNN) and Bi-LSTM, overcoming the limitations of CNN and RNN. The MCNN module collected shallow local semantic information flexibly, and a MIX attention method improved important feature extraction. Experimental findings showed higher classification accuracy than benchmark datasets. Limitations included a greater network size and a longer training time. The authors' CBM model helps to advance text categorization.

In recent research by Wang and Hu [23], a comparison between Support Vector Machine (SVM) and Least Squares SVM (LS-SVM) for regression tasks revealed insights into their performance and computational characteristics. This analysis parallels the pioneering work of Kici *et al.* [24], who explored a BERT-based transfer learning approach for text classification on software requirements specifications, showcasing the evolving landscape of machine learning techniques across diverse domains.

Xu *et al.* [25] propose a novel approach to product requirement development leveraging multi-layer heterogeneous networks. This method addresses the complexity of product requirements by integrating diverse data sources and modeling relationships across different layers. By applying advanced network analysis techniques, the authors offer insights into enhancing the efficiency and effectiveness of the product development process.

III. Materials and Methods

Various Machine Learning (ML) libraries utilized in the model are:

- Beautiful Soap (BS)
- NumPy (Numerical Python)
- Scikit learn (Sklearn)
- Imbalanced learn (imblearn)
- Keras
- Early Stopping Callback
- Model Checkpoint
- Dropout layer
- Activation functions

A. Overview of Dataset

To conduct this study, we made use of the PROMISE_exp dataset. An augmentation of the original PROMISE dataset, this repository aims to promote software engineering prediction models that are repeatable, verifiable, refutable, and susceptible to refinement. The goal of this repository is to promote these models. The UCI Machine Learning Repository served as a source of inspiration for this. In the initial repository, there is an inventory of 255 Functional Requirements (FRs) and 370 non-functional requirements (NFRs), with the latter being further categorized into 11 different NFR classes. The collection has been pre-categorized. The PROMISE repository can be found in the link—PROMISE Software Engineering Repository (uottawa.ca). Fig. 1 shows the distribution of classes of the dataset. There are a total of 969 requirements that are included in the expanded version of PROMISE, which are stated in Table I.

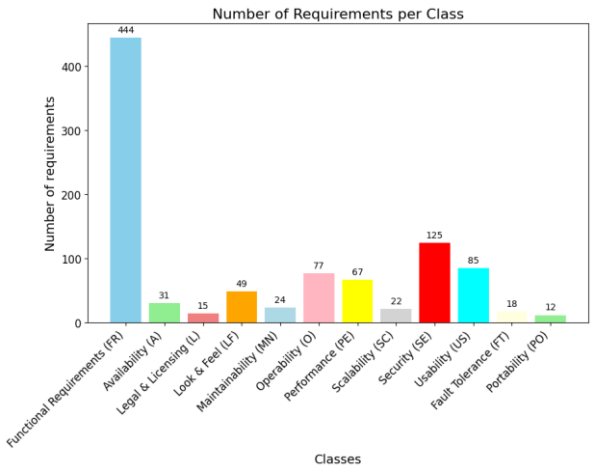

Fig. 1. PROMISE_exp dataset.

B. Methodology

1) Oversampling

Usually, in any real data set, there are always some degrees of imbalance between classes. If the level of imbalance is relatively low, there should not be any big impact on ML model performance. In our dataset, there is high degree of imbalance between requirements categories (classes). This issue is common in the requirements classification field, where the number of

NFR sentences is always very small compared to FR and other contexts that don't include requirements. Furthermore, the number of NFR categories in the same document is various. This issue led us to use oversampling. Fig. 2 shows the class distribution of the dataset after oversampling.

TABLE I. NUMBER OF REQUIREMENTS PER LABEL (PROMISE_EXP DATASET)

Requirement Sentence Type	Symbolic Name	Number
Functional Requirement	FR	444
Availability	A	31
Legal & Licensing	L	15
Look & Feel	LF	49
Maintainability	MN	24
Operability	O	77
Performance	PE	67
Scalability	SC	22
Security	SE	125
Usability	US	85
Fault Tolerance	FT	18
Portability	PO	12
Total		**969**

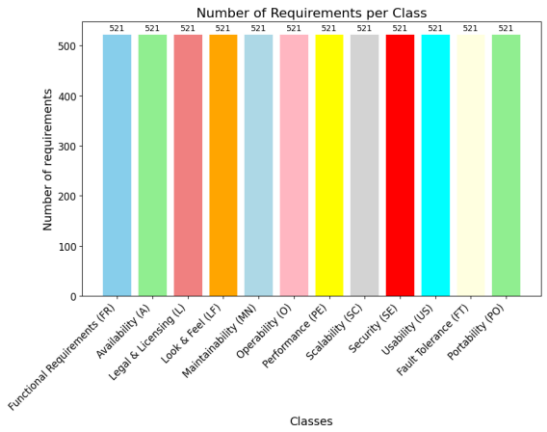

Fig. 2. Oversampled dataset.

2) Dataset preprocessing

To make the process of processing and extracting features easier, the current document requires that it be segmented into smaller portions. In addition, the requirement statements must be revised to exclude lines or paragraphs from these documents because they are not connected to the criteria that have been given. The task was carried out in a sequential fashion, consisting of three distinct stages: tokenization, data cleaning, and normalization. This was done to accomplish the goal mentioned above. In tokenization, the requirements document is broken up into smaller segments.

This process is also called data preparation. The requirements document in this process is broken into paragraphs, and the paragraph into sentences. In order to establish the limits of the phrase, we made use of a selection of criteria that had been established beforehand. It is not possible to obtain any information. These criteria included the utilization of a capital letter in the initial part of the sentence, and the addition of punctuation characters that include a full stop, questioning mark, or exclamation point at the conclusion of the phrase.

The objective of the data cleaning process is to clean all irrelevant tokens from requirement sentences that may undermine the performance of our model. We accomplished this task in three steps. The first step is punctuation removal, in this step, all punctuation marks such as stops, question marks, commas, colons, etc. are removed from the requirement sentences. Where the semantic meaning in the text is based on the basic words. The second step is stop-word removal, in this step, all high-frequency words, such as ("they", "them", "their", "you", "should", "from", etc.) don't add any essential information to the requirement sentence. The last step of the data cleaning task is non-alphabetic tokens removal that didn't contain useful information. In this step, all irrelevant data are removed including punctuation, stop-words, and non-alphabetic tokens.

In normalization process, we aimed to convert all the words to a more uniform sequence by transforming it to a common base form. In this task, we improve text modelling and matching. This task is applied on the words level by three steps: case folding, Parts of Speech (POS) tagging and Lemmatization. Fig. 3 visualizes every step of the preprocessing the dataset.

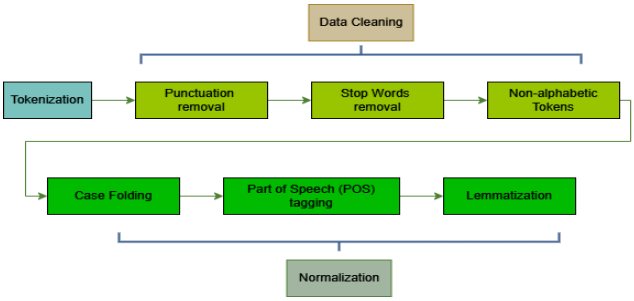

Fig. 3. Data preparation.

a) Word embedding

The process begins with the creation of an empty embedding matrix. This matrix serves as the foundation for incorporating both randomly initialized values and pre-trained word embeddings.

Its dimensions are determined by the vocabulary size and the specified embedding dimension. Subsequently, each word in the vocabulary undergoes evaluation. For words with corresponding pre-trained embedding vectors, these vectors replace the initially random values in the embedding matrix. This step ensures that words with available pre-trained embeddings contribute knowledge to the model.

The embedding matrix is then integrated into a Keras Embedding layer. This layer is configured with essential parameters, including the vocabulary size, embedding dimension, and the maximum sequence length of input data. Importantly, the layer is set to be trainable, allowing its weights to be updated during the subsequent training phase.

b) Bi-LSTM integration with a dense layer

Bi-LSTM layer is followed by a Dense layer within a neural network. This combination enhances the model's

ability to capture bidirectional context and process sequential information.

In this segment, a Bidirectional LSTM layer is applied to the embedded sequences. The Bidirectional wrapper enables the LSTM unit to process input sequences in both forward and backward directions, enriching the model's understanding of temporal dependencies. The parameter 100 denotes the number of units (or neurons) in the LSTM layer.

c) Dropout layer after dense layer

Building upon the output of the Bidirectional LSTM layer, a Dense layer is added with 50 neurons and a Rectified Linear Unit (ReLU) activation function. This layer enhances feature extraction and prepares the data for subsequent processing.

Following the Dense layer, a Dropout layer is introduced. The dropout rate, set at 0.5 in this instance, determines the fraction of randomly selected neurons to be deactivated during each training iteration. This randomness aids in preventing overfitting and promotes the learning of more robust representations.

d) Integrating SVM output layer

After the Dropout layer, a Dense layer is added to produce the SVM output. The number of neurons is determined by the length of Macronum, and the activation function is set to "linear" to maintain the linearity of the SVM output. The Keras Model is then constructed, taking the input sequence, and connecting it to the SVM output layer. This defines the end-to-end architecture of the neural network, with the capability to handle SVM-specific tasks.

e) Model training and evaluation

We use a technique called stratified K-fold cross-validation to train the hybrid model. This technique ensures that each category has an appearance in both the training set and the validation set in a manner that is proportional to the number of times it appears in the dataset. This rigorous training regimen is augmented with early stopping to prevent overfitting and ModelCheckpoint to save the best performing model. To provide a full understanding of the model's performance, it is evaluated using a number of different measures, including accuracy, precision, recall, and F1-Score. In addition to this, confusion matrices offer vital information regarding the quality of the model in accurately classifying examples from each category. Fig. 4 represents the overall flowchart of conducting experiment of the proposed model.

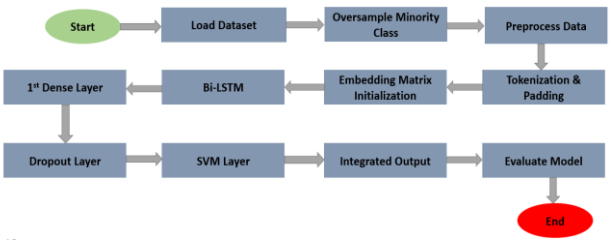

Fig. 4. Flowchart of the working of hybrid model.

IV. Result and Discussion

This experiment is performed using HP 15-DY1XXX which has a RAM of 8 GB, 256 GB SSD, and Windows 10 64-bit OS. For utilizing and training of the proposed model, Google Colab is used with T4 GPU. Google colab gives a systematic and proficient environment for running the model. The suggested model performed the preparation, validation, and testing procedures using the PROMISE_exp dataset. Once the training and validation phases were complete, the model was able to recognize the testing data, which consisted of 969 requirements distributed over 12 classes, with an accuracy of 98.79%.

TABLE II. Comparison with Existing Work

Referencing Work	Dataset (Classes)	Accuracy
Zhang *et al.* [6]	PROMISE FR/NFR, 17 sub-classes	95.7% (binary), 93.4% (multi-class)
Sun *et al.* [17]	DOORS Next Gen, PROMISE-NFR FR/NFR	90.5% (DOORS), 87.8% (PROMISE)
Khan *et al.* [20]	PROMISE FR/NFR	97.3% (LSTM), 96.1% (GRU)
Muhammad *et al.* [2]	PROMISE FR/NFR, 16 sub-classes	90.7% (Bi-LSTM)
Li *et al.* [15]	PROMISE FR/NFR	88.9% (Bi-LSTM)
Rahman *et al.* [2]	PROMISE FR/NFR	87.2% (LSTM)
Sun *et al.* [16]	IEEE Standard 820 FR/NFR Performance	85.4% (Bi-LSTM)
Moreno *et al.* [18]	JIRA, GitHub, Stack Overflow Bug reports, Feature requests	83.1% (LSTM)
Guzman *et al.* [7]	NASA requirements documents Functional, Non-functional, Interface	81.8% (LSTM)
Wang *et al.* [19]	IEEE Transactions on Software Engineering FR/NFR, 10 sub-classes	96.2% (Bi-LSTM + CNN)
Xu *et al.* [20]	IEEE Transactions on Neural Networks and Learning Systems FR/NFR, 12 sub-classes	96.8% (Bi-LSTM + Transformer Encoder)
Proposed Model	Promise_exp FR/NFR, 12 sub-classes	98.79% (Bi-LSTM+SVM)

Table II demonstrates the similar works of previous authors with their used dataset and achievements. In terms of categorization, the model did well consider the large number of requirement classes.

To obtain high accuracy and effectively classify required sentences, we used an SVM-equipped Bidirectional LSTM in the output layer. The final classification is aided by SVM's capacity to handle high-dimensional data and non-linear relationships, while Bidirectional LSTM helps the model efficiently collect contextual information. The integration of these models was evaluated using a 10-fold cross-validation, offering a thorough examination of their predictive power and guaranteeing a reliable performance evaluation.

SVM was strategically chosen for the output layer because of its reputation for resilience, especially in situations requiring the delineation of intricate decision boundaries. To improve the classification task's overall performance, we combined the benefits of SVM's efficiency with deep learning's strengths. Here, Fig. 5 shows the training and validation loss and accuracy curves. It can be seen that the training and validation loss decreases and stabilizes, and the training and validation accuracy increases and stabilizes, which is the proof of ideal scenarios of model training, and proves that the model is not under or overfitted. Fig. 6 is a visual representation of what early stopping callback does in a machine learning model. Early Stopping callback function prevents overfitting by stopping the learning early if performance degrades, saves resources by avoiding unnecessary epochs, and ensures the model performs well to unseen new data.

Fig. 5. Training validation loss accuracy curve.

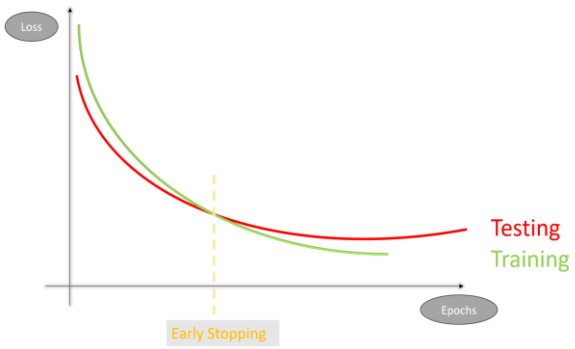

Fig. 6. Early stopping callback.

Table III summarizes the machine learning model's performance characteristic throughout ten folds. The measures are Accuracy, Precision, Recall, and F1-Score. The model is highly consistent across several folds, with all accuracy values greater than 0.98. This implies a consistent performance. The only noticeable difference is in Fold 2, in which the recall and hence the F1-Score are lower than in the other folds. This shows that, though the model is usually robust, there may be unique instances or traits in Fold 2 that pose a slight challenge to the model. Despite this, the model's performance is consistently reliable and effective.

TABLE III. EFFECT OF SVM LAYER IN EACH FOLD

Fold	Accuracy	Precision	Recall	F1-Score
1	0.9848	0.9919	0.9924	0.9920
2	0.9840	0.9832	0.9361	0.9366
3	0.9880	0.9884	0.9880	0.9878
4	0.9936	0.9936	0.9936	0.9936
5	0.9848	0.9849	0.9848	0.9845
6	0.9896	0.9898	0.9896	0.9895
7	0.9912	0.9914	0.9912	0.9911
8	0.9872	0.9847	0.9840	0.9836
9	0.9880	0.9883	0.9880	0.9878
10	0.9912	0.9903	0.9904	0.9904

Bi-LSTM forward backward calculation equations:

$$hf = (Wf \times I + bf + hf)\sigma \quad (1)$$

$$hb = (hb + Wb \times I + bb)\sigma \quad (2)$$

$$o = (hfWf + b + hbWb) \quad (3)$$

V. CONCLUSION

To sum up, the objective of our study was to classify requirement sentences. The hybrid classification model showed promising results in successfully classifying requirement sentences by combining the strengths of SVM and Bi-LSTM. By using Bi-LSTM, the model was able to find out the language structures in requirement sentences on a deeper level by capturing dependencies and contextual information within the input sequences. SVM's complimentary nature and capacity to determine boundaries in high-dimensional space added to the classification system's overall resilience and generalization. By verifying the model's performance by comparing it with other models and existing works, the Bi-LSTM-SVM helped to ensure the model's dependability and reduce the chance of overfitting. The suggested hybrid approach's stability and effectiveness are highlighted by the consistent performance metrics that were acquired across several folds in the 10-fold cross validation.

The performance of the hybrid model could be improved and optimized by conducting further investigations into other feature engineering techniques, neural network topologies, and the integration of domain-specific data. However, our current work provides a solid foundation for the development of necessary sentence categorization algorithms and shows the promise of hybrid models in addressing the complexities involved in this challenging endeavor.

Beyond software engineering, the implications of this work extend to other fields with complex text classification challenges, such as natural language processing and information retrieval. By addressing the limitations of individual models, hybrid approaches like ours offer greater flexibility and adaptability, paving the way for more accurate and nuanced text analysis across diverse domains. While recognizing the potential challenges and limitations built-in in any model, we remain confident that our research represents a considerable improvement in addressing the complexities

of requirement classification. By continuously refining and expanding this approach, we can unlock new possibilities for efficient and effective software development, while also contributing to the advancement of broader text analysis methodologies.

The study was conducted on an existing dataset which has a limited number of classes. In future, we want to work on our own big dataset which will be extracted from an SRS document and be further analyzed.

Abbreviations and Acronums

ML	Machine Learning
DL	Deep Learning
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
Bi-LSTM	Bidirectional Long Short-Term Memory
SVM	Support Vector Machine
NB	Naive Bayes
LR	Logistic Regression
KNN	K-Nearest Neighbors
BS	Beautiful Soup
Numpy	Numerical Python
ReLU	Rectified Linear Unit
FR	Functional Requirement
NFR	Non-Functional Requirement
A	Availability
L	Legal & Licensing
LF	Look & Feel
MN	Maintainability
O	Operability
PE	Performance
SC	Scalability
SE	Security
US	Usability
FT	Fault Tolerance
PO	Portability

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Mahmuda and Nazneen conceived the presented idea. Manhmuda, Nazneen and Nasrin designed the model and the computational framework and analyzed the data. Mahmuda, Nazneen, and Nasrin carried out the implementation and performed the calculations. Mahmuda, Nazneen, and Nasrin wrote the manuscript with input from all authors. Anzum gave feedback about the existing work. Afrina and Mazumder conceived the study and were in charge of overall direction and planning. All authors had approved the final version.

Acknowledgment

Our sincere and honest gratitude goes to our supervisor, Rashed Mazumder, Associate Professor, at the Institute of Information Technology, Jahangirnagar University, for granting us the opportunity to study in this discipline and for his invaluable guidance throughout the research process. We were greatly inspired and motivated by his vitality, foresight, integrity, and enthusiasm. His instruction has encompassed the techniques required to carry out the research and ensure the utmost transparency in the work. Conducting research under his direction was an honor and a tremendous privilege.

References

[1] X. Zhou, X. Wan, and J. Xiao, "Attention-based LSTM network for cross-lingual sentiment classification," in *Proc. the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 247–256.

[2] M. A. Rahman, M. A. Haque, M. N. A. Tawhid, and M. S. Siddik, "Classifying non-functional requirements using rnn variants for quality software development," in *Proc. the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation*, Aug. 2019.

[3] K. Rahman, A. Ghani, R. Ahmad, and S. H. Sajjad, "Hybrid deep learning approach for nonfunctional software requirements classifications," in *Proc. the 2023 International Conference on Communication, Computing and Digital Systems (CCODE)*, IEEE, 2023.

[4] F. Khayashi, B. Jamasb, R. Akbari, and P. Shamsinejadbabaki, "Deep learning methods for software requirement classification: A performance study on the pure dataset," arXiv preprint, arXiv:2211.05286, 2022.

[5] F. Yucalar, "Developing an advanced software requirements classification model using BERT: An empirical evaluation study on newly generated Turkish data," *Applied Sciences*, vol. 13, no. 20, 11127, 2023.

[6] K. Kaur and P. Kaur, "A self-attention based bidirectional-RNN deep model for requirements classification," *Journal of Software: Evolution and Process*, February 2022.

[7] N. Rahimi, F. Eassa, and L. Elrefaei, "One- and two-phase software requirement classification using ensemble deep learning," *Entropy*, vol. 23, 1264, 2021.

[8] A. M. Ali and N. N. Saleem, "Classification of software systems attributes based on quality factors using linguistic knowledge and machine learning: A review," *Journal of Education and Science*, vol. 31, pp. 66–90, 2022.

[9] M. Sabir, "Optimisation method for training deep neural networks in classification of non-functional requirements," Doctoral dissertation, London South Bank University, UK, September 2022.

[10] S. Saratha and S. Mukherjee, "A novel approach for improving the accuracy using word embedding on deep neural networks for software requirements classification," Research Square preprint, 2023. https://doi.org/10.21203/rs.3.rs-2742342/v1

[11] F. Baskoro, R. A. Andrahsmara, B. R. P. Darnoto, and Y. A. Tofan, "A systematic comparison of software requirements classification," *IPTEK The Journal for Technology and Science*, vol. 32, 2021.

[12] G. Li, C. Zheng, M. Li, and H. Wang, "Automatic requirements classification based on graph attention network," *IEEE Access*, vol. 10, pp. 30080–30090, March 2022.

[13] B. Jang, M. Kim, G. Harerimana, S.-U. Kang, and J. W. Kim, "Bi-LSTM model to increase accuracy in text classification: Combining word2vec CNN and attention mechanism," *Applied Sciences*, vol. 10, 5841, 2020.

[14] K. Kaur and P. Kaur, "Improving BERT model for requirements classification by bidirectional LSTM-CNN deep model," *Computers and Electrical Engineering*, vol. 108, 108699, 2023.

[15] B. Li and X. Nong, "Automatically classifying non-functional requirements using deep neural network," *Pattern Recognition*, vol. 132, 108948, 2022.

[16] S. Tiun, U. A. Mokhtar, S. H. Bakar, and S. Saad, "Classification of functional and non-functional requirement in software requirement using word2vec and fast text," *Journal of Physics: Conference Series*, vol. 1529, 042077, 2020.

[17] H. Huan, J. Yan, Y. Xie, Y. Chen, P. Li, and R. Zhu, "Feature-enhanced nonequilibrium bidirectional long short-term memory model for Chinese text classification," *IEEE Access*, vol. 8, pp. 199629–199637, 2020.

[18] K. Rahman, A. Ghani, S. Misra, and A. U. Rahman, "A deep learning framework for non-functional requirement classification," *Scientific Reports*, vol. 14, no. 1, 3216, 2024.
[19] A. E. Yahya, A. Gharbi, W. M. Yafooz, and A. Al-Dhaqm, "A novel hybrid deep learning model for detecting and classifying non-functional requirements of mobile apps issues," *Electronics*, vol. 12, no. 5, 1258, 2023.
[20] D. Zheng, "Transfer learning-based English translation text classification in a multimedia network environment," *PeerJ Computer Science*, vol. 10, e1842, 2024.
[21] G. Tian, Q. Wang, Y. Zhao, L. Guo, Z. Sun, and L. Lv, "Smart contract classification with a BI-LSTM based approach," *IEEE Access*, vol. 8, pp. 43806–43816, 2020.
[22] H. Huan, Z. Guo, T. Cai, and Z. He, "A text classification method based on a convolutional and bidirectional long short-term memory model," *Connection Science*, vol. 34, no. 1, pp. 2108–2124, 2022.
[23] H. Wang and D. Hu, "Comparison of SVM and LS-SVM for regression," in *Proc. 2005 International Conference on Neural Networks and Brain*, IEEE, 2005, pp. 279–283.
[24] D. Kici, G. Malik, M. Cevik, D. Parikh, and A. Basar, "A BERT-based transfer learning approach to text classification on software requirements specifications," in *Proc. Canadian Conference on AI*, 2021, vol. 1, 042077.
[25] X. Xu, Y. Dou, W. Ouyang, J. Jiang, K. Yang, and Y. Tan, "A product requirement development method based on multi-layer heterogeneous networks," *Advanced Engineering Informatics*, vol. 58, 102184, 2023.