BERT Fine-Tuning for Software Requirements Classification: Impact of Model Components and Dataset Size

Safaa Eltahier , Omer Dawood , Imtithal Saeed *

Posted Date: 20 October 2025

doi: 10.20944/preprints202510.1480.v1

Keywords: requirements engineering; software requirements classification; functional requirements; non-functional requirements; BERT fine-tuning

 doi:10.20944/preprints202510.1480.v1

Article

BERT Fine-Tuning for Software Requirements Classification: Impact of Model Components and Dataset Size

Safaa Eltahier [1,2], Omer Dawood [3] and Imtithal Saeed [4,*]

[1] Software Engineering Department, College of Computer Engineering and Sciences Prince Sattam Bin Abdulaziz University,Al-kharj, KSA
[2] College of Computer Science and Information Technology, Sudan University of Science and Technology, Khartoum, Sudan
[3] Computer Engineering and Information Department, College of Engineering in Wadi Aldawasir Prince Sattam Bin Abdulaziz University Wadi Aldawasir, KSA
[4] Information Systems Department, College of Computer Engineering and Sciences Prince Sattam Bin Abdulaziz University Al-kharj, KSA
* Correspondence: i.mohammedzein@psau.edu.sa

Abstract

Recent advances in natural language processing (NLP) have enabled the automation of Software Requirements Classification (SRC), particularly through fine-tuning models such as Bidirectional Encoder Representations from Transformers (BERT). While BERT-based models have shown promising results, the impact of hyperparameter sensitivity and dataset size on SRC performance remains underexplored. To address this gap, we present three main contributions: (1) the development and evaluation of BERT fine-tuning for SRC, with emphasis on the effects of key hyperparameters and dataset size; (2) comprehensive experiments to analyze the influence of individual hyperparameters to identify optimal configurations for robust and efficient performance; and (3) controlled experiments highlighting the critical factors that affect the fine-tuning outcomes in SRC, particularly dataset size and hyperparameter sensitivity. Our approach was assessed on two datasets: the PROMISE NFR dataset and FR_NFR, a tailored dataset for the SRC task. The proposed method outperformed baseline models, achieving an average F1-score of 0.99 on PROMISE and 0.97 on FR_NFR. These findings provide empirical evidence on optimization strategies for BERT-based requirements classification and offer practical guidance to software engineering practitioners.

Keywords: requirements engineering; software requirements classification; functional requirements; non-functional requirements; BERT fine-tuning

1. Introduction

Software requirements define what a system should do, specifying its expected behavior, essential properties, and constraints on development, such as performance, security, or regulatory compliance. To ensure these requirements are correctly captured and maintained, requirements engineering (RE) —subset of Requirements Engineering— provides a structured approach for eliciting, analyzing, documenting, and managing them throughout the software development lifecycle. By systematically handling requirements, RE helps ensure that the final system meets stakeholders' needs, reduces development risks, and improves overall software quality (Sommerville and Sawyer, 1999).Software requirements are commonly classified into two categories: functional requirements (FRs) and non-functional requirements (NFRs). FRs specify the system's expected behavior in particular situations and define what developers must implement, whereas NFRs

 doi:10.20944/preprints202510.1480.v1

describe the properties, quality attributes, or constraints that the system must satisfy. The requirements analysis process—a critical phase of the Software Development Life Cycle (SDLC)—focuses on identifying, refining, and structuring both FRs and NFRs from broader system information. Developers rely on these requirements to design solutions that deliver the intended functionality, while testers use them to ensure that the implemented system satisfies both functional specifications and quality expectations. Together, FRs and NFRs provide the foundation for software development and validation, helping to ensure that the final system meets stakeholder needs and adheres to required standards. (Wiegers et al., 2013). However, manually extracting FRs and NFRs from requirement documents is error-prone, labor-intensive, and time-consuming. Consequently, automating this process can improve efficiency, reduce human errors, and ensure more consistent requirement analysis.

Recently, function has become increasingly associated with Natural Language Processing (NLP), since requirements are typically written in natural language by analysts or domain experts. NLP techniques enable the automated processing and interpretation of these textual requirements, but they also introduce the challenge of ensuring classification is both efficient and accurate. Within this NLP-based framing, SRC can be formulated as a binary classification problem, distinguishing between FRs and non-functional requirements NFRs. Alternatively, it can be modeled as a multi-class classification problem, where NFRs are further categorized into specific quality attributes such as security, usability, interoperability, and others.

Researchers have proposed several approaches based on Machine Learning (ML), Deep Learni ng (DL), and Transfer Learning (TL) techniques(Kaur and Kaur, 2024). Traditional ML have achieved good accuracy in classification task; however, there are issues with this approach, such as dependence on feature engineering and a lack of contextual understanding. Other DL methods such as CNN classifiers resulted in high performance particularly high precision and recall for NFR classification(Airlangga, 2024).However CNN requires large labelled datasets.

Recent advances in NLP have significantly improved the field of requirements classification, particularly with the use of pre-trained language models like BERT, and its equivalents. Numerous studies such as (Kaur and Kaur, 2023) (Kaur and Kaur, 2023) (Luo et al., 2022) have examined the application of these models to automate the classification of software requirements. However, the optimal configuration of BERT for SRC tasks remains unclear, with limited understanding of which factors most significantly impact the performance during fine-tuning. This gap in knowledge often leads to suboptimal model configurations and inconsistent results across different requirements datasets.

1.1. Research Problem

Although BERT has shown remarkable effectiveness across a wide range of NLP tasks, its application to SRC has not yet been comprehensively explored. In particular, there is limited understanding of the factors that most significantly influence performance improvements. Researchers often face challenges such as:

- Assessing the sensitivity of model performance to different hyperparameter settings
- Identifying the key components of the fine-tuning process that contribute most to classification accuracy
- Determining the minimum and optimal dataset sizes required for effective fine-tuning

1.2. Research Questions

To address these gaps, this study is guided by the following research questions (RQs):

- RQ1: Which hyperparameters have the most significant impact on BERT fine-tuning effectiveness for the SRC task?
- RQ2: How does dataset size affect BERT performance in the SRC task?
- RQ3: What is the optimal combination of dataset size and hyperparameter settings for maximizing performance in SRC tasks?

 doi:10.20944/preprints202510.1480.v1

1.3. Research Contribution

This research advances the field of **RE** in the following ways:

1. **Optimally tuned BERT model for SRC**

We develop and evaluate a systematically fine-tuned BERT model specifically tailored for SRC. Unlike prior studies that applied BERT without a deep investigation into fine-tuning, this work identifies the optimal hyperparameter configurations that significantly impact classification performance. By addressing this gap, our approach achieves superior results compared to state-of-the-art methods.

2. **Empirical analysis of fine-tuning factors**

We conduct a systematic empirical analysis of the influence of hyperparameters and dataset size on BERT's performance in SRC. Through controlled experiments and ablation studies, we quantify the relative importance of individual components in the fine-tuning process. The findings not only provide practical guidelines for researchers and practitioners on effective model setup and resource utilization but also contribute to advancing transformer-based approaches in RE.

Finally, the remainder of the paper is organized as follows: Section 2 reviews related work, highlighting recent advancements in NLP models for requirements classification. Section 3 describes the proposed methodology, including the fine-tuning process, experimental setup, and datasets. Section 4 presents and discusses the results, with comparisons against existing approaches. Section 5 outlines the research limitations and suggestions for future work. Section 6 concludes the paper by summarizing the key contributions, outlining limitations, and suggesting future research directions in applying large language models (LLMs) to RE.

2. Related Work

Recently, the automation of the SRC task has gained considerable attention in the RE research community. It plays a crucial role in managing large-scale datasets by significantly reducing the manual effort required from requirements engineers. To address the challenges of SRC, researchers have proposed a range of approaches, including those based on ML, DL, and TL. This section provides a review of the existing literature on SRC, with a focus on the key contributions, methodologies, and findings reported in prior studies.

For example, (Kurtanović and Maalej, 2017) developed a Support Vector Machine (SVM) classifier that leveraged metadata, lexical, and syntactical features, achieving precision and recall of up to 92% in automatically distinguishing FRs from NFRs. Similarly, (Li et al., 2022) applied ML methods, following the approach described by (EzzatiKarami and Madhavji, 2021), to classify software requirements into functional and non-functional categories. Using TF-IDF for text feature representation, their experiments showed that Naïve Bayes achieved the best performance, with an F1-score of 86% for functional requirements and 90% for non-functional requirements.

Abad et al. (2017) conducted a comparative study of several machine learning algorithms for classifying NFRs, including Latent Dirichlet Allocation (LDA), Biterm Topic Model (BTM), K-means, Naïve Bayes, and a hybrid approach combining hierarchical clustering with K-means. Their experiments were performed on both raw and preprocessed versions of the PROMISE NFR dataset. The preprocessing, which involved rule-based techniques and domain-specific dictionaries, significantly enhanced classification performance, yielding a weighted average F1-score of 94% on the processed data. However, a notable limitation of their approach is the reliance on manual preprocessing, which may be tailored to specific datasets and thus limit generalizability.

Additionally, several studies have applied traditional machine learning techniques to the SRC task, specifically for identifying subclasses of non-functional requirements (NFRs). Notable examples include (Haque et al., 2019) (Shreda and Hanani, 2021) and (Jindal et al., 2021), who explored various algorithms to enhance NFR classification accuracy. However, despite promising results, the generalizability of these approaches across different domains remains unestablished, as

 doi:10.20944/preprints202510.1480.v1

highlighted by (Airlangga, 2024), raising concerns about their applicability in diverse real-world settings.

Research in NFR classification has gradually shifted toward deep learning techniques, offering improved performance over traditional machine learning approaches. For instance, (Navarro-Almanza et al., 2017) employed a CNN to classify NFR subclasses in the PROMISE NFR dataset without relying on handcrafted features. Their model outperformed traditional methods, demonstrating the effectiveness of deep learning in capturing complex patterns. Similarly, (Baker et al., 2019) applied both CNN and artificial neural networks (ANN) to classify NFRs into five categories across two datasets, achieving F-scores ranging from 82% to 92% with the CNN model. However, CNNs typically require large amounts of labeled training data, which can be a limitation in many domains (Airlangga, 2024). To address this, recent approaches have increasingly adopted transfer learning and pretrained models. One prominent example is BERT, which combines bidirectional transformers with transfer learning to deliver state-of-the-art performance in various language tasks (Kici et al., 2021).

(Li et al., 2022) proposed a novel method for the SRC task by incorporating sentence structure and syntactic information. They constructed dependency parse trees and utilized a Graph Attention Network (GAT) to extract both implicit structural features and syntactic features from the requirements. Their approach involved four graph-construction strategies combined with GAT, and the dependency-graph-based method achieved precision, recall, and F1-scores exceeding 90% for both functional and non-functional requirements classification. To further enhance performance, they integrated BERT with the graph-construction method, resulting in a weighted average F1-score of 94%, comparable to the results reported by (Abad et al., 2017). This integration highlights the effectiveness of combining syntactic modelling with pretrained language representations in SRC tasks.

(Kaur and Kaur, 2023) proposed a BERT-CNN-based approach for the software requirements classification (SRC) task. To evaluate its effectiveness, they conducted experiments on the PROMISE dataset, focusing on multi-class classification across four NFR categories: Operability, Performance, Security, and Usability. Their results showed that the BERT-CNN model outperformed the baseline BERT model, demonstrating improved accuracy in distinguishing between NFR subclasses.

Derya et al. (Kici et al., 2021) evaluated the effectiveness of the DistilBERT transformer as a transfer learning model for multi-class text classification on software requirements documents. Their findings showed that DistilBERT significantly outperformed traditional RNN-based models.

(Luo et al., 2022) introduced a prompt-based learning approach for the SRC task using a BERT-based pretrained language model called PRCBERT. They conducted experiments on two small-scale software requirements datasets—PROMISE and NFR-Review—and found that PRCBERT outperformed both NoRBERT and MLM-BERT models in classification performance. Similarly, (Hey et al., 2020a) fine-tuned BERT for the software language classification (SLC) task and achieved F1-scores of up to 94% for binary classification on the PROMISE NFR dataset. To optimize memory usage and training efficiency, they limited the maximum sequence length to 128 for base models and 50 for larger models. They also investigated the impact of training epochs, concluding that 10–32 epochs were optimal for binary classification, while 10–64 epochs yielded better results for multiclass settings.

A few semi-supervised approaches have also been explored for the SRC task. For example, (Airlangga, 2024) proposed a framework that integrates Generative Adversarial Networks (GANs) with a BERT-based model to reduce reliance on large annotated datasets, making the approach more suitable for real-world applications. However, the limitations and broader applicability of such semi-supervised methods remain underexplored. According to (Kaur and Kaur, 2024), transfer learning (TL)-based approaches consistently achieve higher accuracy compared to traditional ML and DL techniques, further highlighting the growing preference for pretrained models in SRC tasks.

Recent studies increasingly highlight BERT's robustness in addressing the SRC task, particularly emphasizing the importance of proper fine-tuning to capture contextual dependencies within

 doi:10.20944/preprints202510.1480.v1

requirement texts. BERT's flexibility in fine-tuning makes it well-suited for modern SRC challenges and enhances performance across domain-specific applications. Despite the widespread use of BERT in SRC research, no existing study has thoroughly investigated the impact of hyperparameter tuning and dataset size on model performance. Therefore, ablation studies in this context remain underexplored, and further research is needed to optimize and improve performance in SRC tasks.

3. Material and method

This section details the methodological steps taken and the implementation specifics, including dataset, model configuration and training setup, to ensure reproducibility and clarity. This section explains the process of fine-tuning BERT for SRC. The experimental setup utilized data from the PROMISE NFR and FR_NFR datasets as input. Prior to feeding the data into BERT, the text was carefully pre-processed and cleaned to ensure suitability for the classification task.

3.1. Datasets

Two datasets were employed in the experimental analysis:

A) PROMISE-NFR Dataset:

The PROMISE Non-Functional Requirements (NFR) dataset is among the most widely used resources in software requirements research. Maintained by the School of Information Technology and Engineering at the University of Ottawa since 2005, it comprises 625 software requirements. These are categorized into 255 FRs and 370 NFRs, with the latter further divided into 11 distinct NFR types. Due to its long-standing availability and structured classification, this dataset remains a benchmark for comparative studies in the field (Kaur and Kaur, 2024).

B) PURE Dataset:

The PURE (Public REquirements) dataset consists of 79 publicly available natural language requirements documents collected from various online sources. It includes a total of 34,268 sentences and supports a range of natural language processing tasks relevant to requirements engineering, such as model synthesis, abstraction detection, and document structure analysis ("PURE," 2017). (Sonali and Thamada, 2024) extracted and processed requirements from the PURE repository and additional open-source software documentation to create the FR_NFR dataset. This refined dataset contains 6,117 requirements, of which 3,964 are functional and 2,153 are non-functional.

Table 1 provides representative examples of functional and non-functional requirements from both datasets. Figure 1 illustrates the distribution of FRs and NFRs in the PROMISE-NFR and FR_NFR datasets, highlighting that the FR_NFR dataset is approximately 9.8 times larger than the PROMISE-NFR dataset.

Table 1. Representative examples of FR and NFR from the PROMISE and FR_NFR datasets.

Requirement	Class	Dataset
"The system shall refresh the display every 60 seconds."	NFR	PROMISE
"The system shall filter data by: Venues and Key Events."	FR	
"The app shall run on a smart phone with Android OS least version 2.3 ."	NFR	FR_NFR_dataset
"User shall be able to add, modify, or remove user data, with changes reflected successfully."	FR	

 doi:10.20944/preprints202510.1480.v1

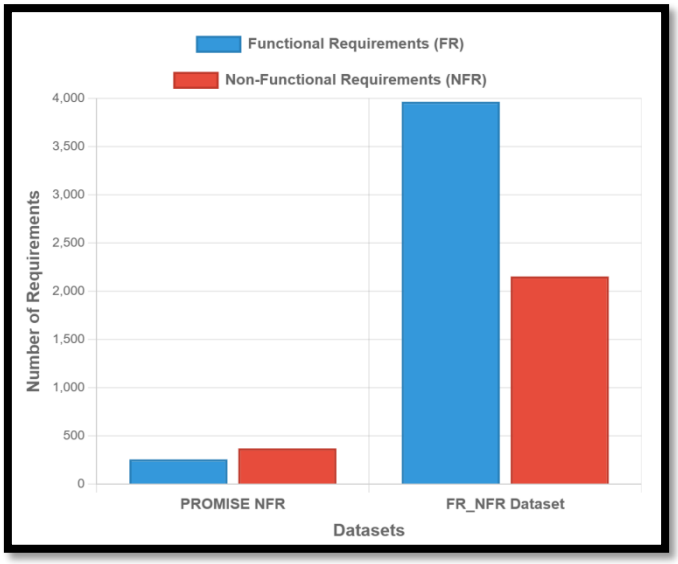


Figure 1. Comparative Distribution of FR and NFR in the PROMISE-NFR and FR_NFR Datasets.

3.2. Model Architecture

The architecture employed in this study is based on BERT (Bidirectional Encoder Representations from Transformers), a multi-layer bidirectional Transformer encoder. All model configurations leverage pre-trained BERT encoders in conjunction with a Multi-Layer Perceptron (MLP) for classification tasks.

BERT processes input sequences using a combination of token embeddings, segment embeddings, and position embeddings, which are summed for each token. The model comprises multiple layers of bidirectional Transformer encoders that utilize self-attention mechanisms and feed-forward networks to capture contextual information from both directions of the input sequence.

A special classification token, [CLS], is prepended to each input sequence. Its final hidden state serves as the aggregate representation for classification. Additionally, the [SEP] token is used to denote sentence boundaries or separate multiple segments within the input. The final hidden state of the [CLS] token is passed through a dropout layer, followed by a fully connected linear layer, and finally a softmax activation function to generate class probabilities. This classification head maps the [CLS] representation to the target label space.

For our experiments, we adopted the BERTbase model due to its balance between performance and computational efficiency compared to BERTlarge. BERTbase consists of:

- 12 Transformer layers (blocks),
- 768-dimensional hidden states,
- 12 self-attention heads,
- approximately 110 million parameters (Devlin et al., 2019; Gardazi et al., 2025).

3.3. Fine-Tuning Process

Fine-tuning the BERT model for SRC involves several critical steps, beginning with pre-processing, which is essential for effective natural language understanding. Raw software requirement texts are first cleaned by removing punctuation, converting all characters to lowercase, eliminating stop words, and applying stemming to reduce words to their base forms. This ensures a standardized input format conducive to accurate classification.

Next, the cleaned text is transformed into a format compatible with BERT. This involves tokenization, which splits the text into subword units. Special tokens such as [CLS] are added at the beginning of each sequence to represent the classification context, while [SEP] tokens are used to denote sentence boundaries or separate segments. Each token is then mapped to its

 doi:10.20944/preprints202510.1480.v1

corresponding index in BERT's vocabulary, resulting in a sequence of input IDs—numerical representations of the tokens.

To maintain uniform input lengths, sequences are padded to a fixed size, and longer sequences are truncated to fit within BERT's maximum supported length. Attention masks are generated to differentiate between actual tokens and padding, allowing the model to focus on meaningful content during training.

The fine-tuning process customizes BERT's pre-trained weights for the SRC task. Instead of relying on default settings, optimal hyperparameters are identified through a systematic ablation study, which evaluates the impact of various configurations on model performance.

Training involves backpropagation, enabling the model to learn task-specific patterns from both the PROMISE-NFR and FR_NFR datasets. The overall research methodology and model development workflow are illustrated in Figure 2. A detailed analysis of hyperparameter effects is provided in Section 4.3, and the optimal configurations for both datasets are summarized in Table 2.

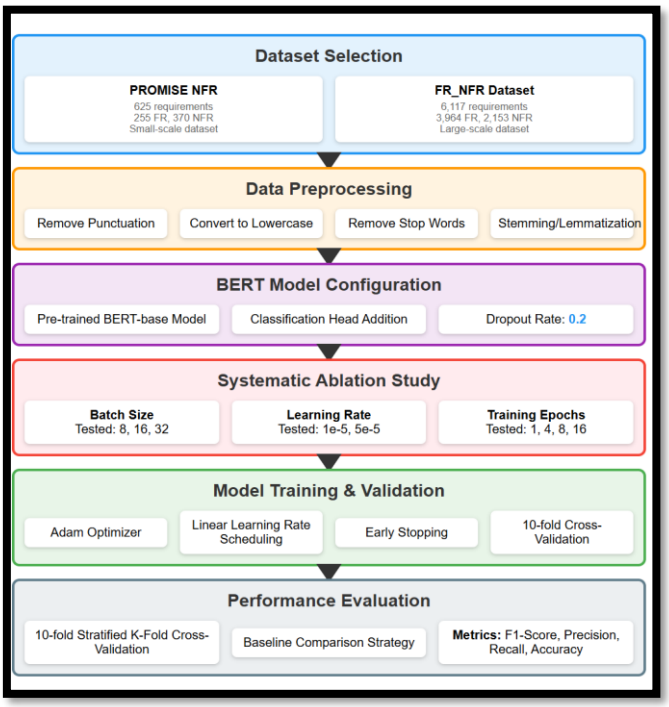


Figure 2. Workflow for Methodology and Fine-Tuned BERT Model Development.

Table 2. Top-performing Hyperparameters for BERT Fine-Tuning on SRC task.

Parameter	PROMISE NFR dataset	FR NFR dataset
Optimizer	Adam	Adam
Max Seq Length	256	256
Dropout Rate	0.2	0.2
Batch Size	8	16
Learning Rate	1e−5	1e−5
Epochs	8	4

For optimization, the model employs the Adaptive Moment Estimation (ADAM) optimizer (Kingma and Ba, 2017), widely recognized for its adaptive learning rate and efficient convergence (Gkouti et al., 2024). The sparse categorical cross-entropy loss function is used to measure the discrepancy between predicted and actual class labels, making it well-suited for multi-class classification tasks.

 doi:10.20944/preprints202510.1480.v1

To ensure consistency and reliability in evaluation, the same dataset splitting methodology used in baseline studies is adopted. Specifically, a 10-fold Stratified K-Fold Cross-Validation technique is implemented to maintain balanced class distributions across all splits, ensuring unbiased performance assessment. The results of this evaluation are presented in Table 3.

4. Results & Discussion

This study investigates the impact of hyperparameter sensitivity and dataset size on the performance of the traditional BERT model when fine-tuned for the SRC task. An ablation analysis approach was adopted to systematically evaluate how different configurations influence model effectiveness. This section presents and discusses the experimental findings in detail.

4.1. Experimental Results

A comprehensive evaluation of the fine-tuned BERT model is provided in this section. The complete source code used for experimentation is publicly available via GitHub [1] ensuring transparency and reproducibility.

To assess the model's performance, we conducted comparative experiments using the PROMISE-NFR dataset, which serves as a benchmark in SRC research. Our results are compared against state-of-the-art baseline models, including both traditional machine learning algorithms (e.g., Support Vector Machines, Naïve Bayes) and deep learning approaches, particularly BERT-based models. These comparisons are summarized in Table 3.

In addition to the PROMISE-NFR dataset, we evaluated our model on the FR_NFR dataset, which is approximately ten times larger. This allowed us to examine the scalability and robustness of the fine-tuned BERT model across datasets of varying sizes.

Table 3. Comparison of methods for classifying software requirements into Functional (F) or Non-Functional (NFR) on the PROMISE NFR dataset using 10-fold cross-validation.

Study	Technique	FR			NFR			F1 Avg
		P	R	F	P	R	F	
(Kurtanović and Maalej, 2017)	SVM (word features)	0.92	0.93	0.93	0.93	0.92	0.92	0.93
(Li et al., 2022)	Naïve Bayes (TF-IDF)	0.80	0.93	0.86	0.95	0.85	0.90	0.88
(Hey et al., 2020b)	BERT classfier	0.92	0.88	0.90	0.92	0.95	0.93	0.92
(Li et al., 2022)	Bert+ GAT	0.94	0.90	0.92	0.94	0.98	0.96	0.94
(Abad et al., 2017)	Processed data	0.90	0.97	0.93	0.98	0.93	0.95	0.94
(Luo et al., 2022)	PRCBERT with RoBERTa-large	0.92	0.95	0.93	0.94	0.96	0.95	0.94
Our model	Bert (fine-tuned)	0.98	0.98	0.98	0.99	0.99	0.99	0.99

To evaluate the performance of both the baseline models and the proposed BERT-based model, we utilized standard classification metrics: accuracy, precision, recall, F1-score, and the confusion

[1] Source Code found at: https://github.com/omercomail/Requirements-BERT

 doi:10.20944/preprints202510.1480.v1

matrix. Accuracy reflects the overall correctness of the model's predictions, while precision measures the proportion of correctly predicted positive instances among all predicted positives. Recall assesses the model's ability to identify actual positive instances, and the F1-score provides a balanced measure by combining precision and recall. The confusion matrix offers a detailed breakdown of prediction outcomes, including true positives, false positives, true negatives, and false negatives, allowing for a deeper understanding of model behavior across different classes.

To ensure robust and unbiased evaluation, we adopted a 10-fold Stratified K-Fold Cross-Validation strategy. This method systematically partitions the dataset into ten equal subsets, training the model on nine subsets and testing on the remaining one. The process is repeated ten times, with each subset serving as the test set once. The final performance metrics are averaged across all folds, providing a reliable estimate of the model's generalization capability.

Similar to the approach used by (Li et al., 2022), we evaluated model performance using the F1-score (A), calculated as the weighted average of individual F1-scores across all requirement categories. This metric facilitates a fair comparison across models, especially when dealing with imbalanced datasets. Our proposed fine-tuned BERT model achieved state-of-the-art results, outperforming existing baseline approaches for the SRC task. Among the baseline models, the highest reported F1-score (A) was 0.94, achieved by (Li et al., 2022), (bad et al., 2017) and (Luo et al., 2022) on the PROMISE-NFR dataset.

4.2. Fine-Tuned BERT Model Results for the SRC Task

This section presents a detailed analysis of the results obtained using our proposed fine-tuned BERT model, evaluated on two datasets of different sizes. For the PROMISE-NFR dataset, the model demonstrated superior performance compared to baseline methods, as shown in Table 3. The corresponding confusion matrix in Figure 3(a) provides a visual representation of classification accuracy across classes, while Table 4(a) presents the full classification report.

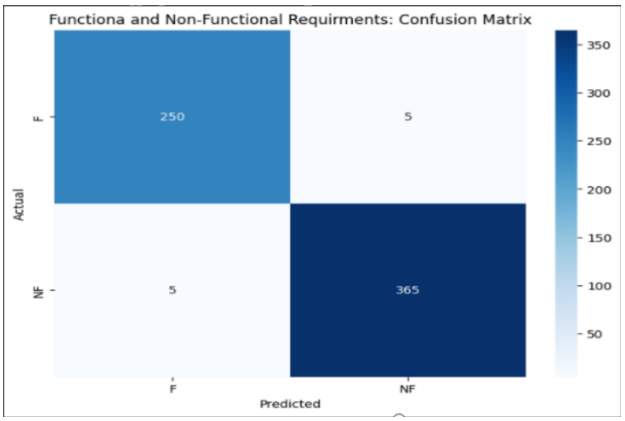
For the FR_NFR dataset, which is significantly larger, the model achieved an F1-score of 0.98 for the NFR class and 0.96 for the FR class using the optimal hyperparameter configuration. These results are detailed in Table 4(b), with the associated confusion matrix shown in Figure 3(b). To the best of our knowledge, no prior studies have reported binary classification results on this dataset, highlighting the novelty and contribution of our work.

Table 4. (a).PROMISE NFR dataset: classification report for SRC task. Table 4(b).FR,NFR dataset: classification report for SRC task.

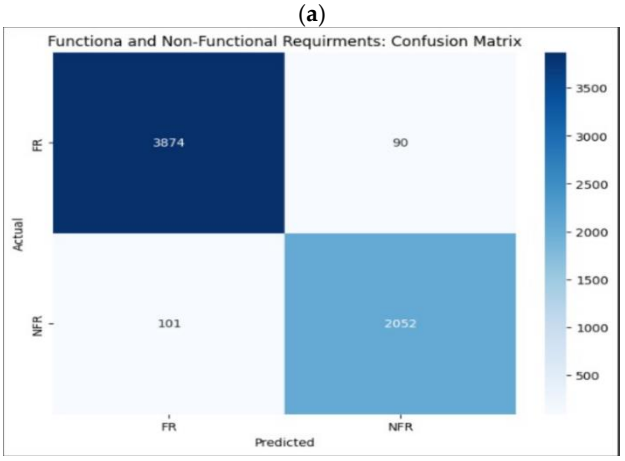
(a)				
	Precision	Recall	F1-Score	Support
FR	**0.98**	**0.98**	**0.98**	**255**
NFR	**0.99**	**0.99**	**0.99**	**370**
Accuracy			**0.98**	**625**
Macro Avg	**0.98**	**0.98**	**0.98**	**625**
Weighted Avg	**0.98**	**0.98**	**0.98**	**625**

(b)				
	Precision	**Recall**	**F1-Score**	**Support**
FR	**0.97**	**0.98**	**0.98**	**3964**
NFR	**0.96**	**0.95**	**0.96**	**2153**
Accuracy			**0.97**	**6117**
Macro Avg	**0.97**	**0.97**	**0.97**	**6117**
Weighted Avg	**0.97**	**0.97**	**0.97**	**6117**

 doi:10.20944/preprints202510.1480.v1



(**a**)



(**b**)

Figure 3. (a). Confusion matrix of SRC task in PROMISE NFR dataset. (b). Confusion matrix of SRC task in FR NFR dataset.

4.3. Hyperparameter Ablation

Fine-tuning BERT for the SRC task requires careful selection of hyperparameters, as they significantly influence model performance and generalization. In this study, we conducted a systematic ablation analysis to identify optimal hyperparameter configurations for fine-tuning BERT across two datasets of different sizes. Key hyperparameters examined include learning rate, batch size, number of training epochs, dropout rate, and weight decay—all of which affect the model's ability to learn effectively and generalize to unseen data.

Following the guidance of (Devlin et al., 2019), we focused primarily on batch size, learning rate, and training epochs, as these are the most critical and task-sensitive parameters in BERT fine-tuning. Our experiments employed a one-at-a-time approach, where each hyperparameter was varied individually while keeping others constant. This method allowed us to isolate and understand the impact of each parameter on model performance.

4.3.1. Effect of Training Epochs on BERT Fine-Tuning Performance

We first investigated the effect of varying the number of training epochs on BERT's performance for the SRC task using the PROMISE-NFR dataset. Given the moderate class imbalance in this dataset, we used the F1-score as the primary evaluation metric and reported performance separately for FRs and NFRs. The F1-score, which balances precision and recall, is particularly suitable for evaluating models under class imbalance conditions.

Figure 4 illustrates the performance trends across different epoch settings. For FRs, the F1-score ranged from 0.96 to 0.99, showing a gradual improvement up to 8 epochs, after which performance slightly declined. A similar pattern was observed for NFRs, with F1-scores also ranging between 0.96 and 0.99, peaking at 8 epochs before experiencing a minor drop. These results suggest that 8

 doi:10.20944/preprints202510.1480.v1

epochs represent an optimal training duration for this task, beyond which the model may begin to overfit.

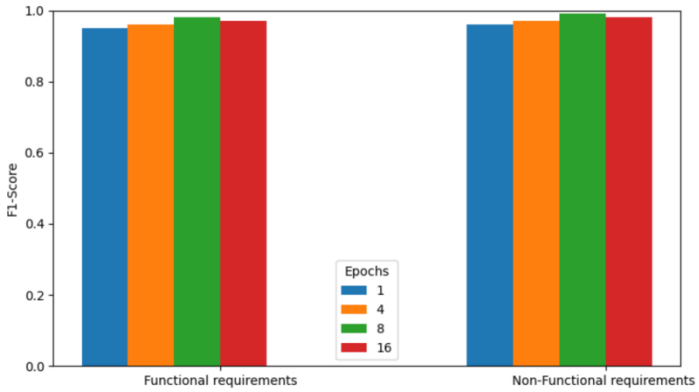


Figure 4. F1-Score across software requirement types and epoch settings in PROMISE NFR dataset.

In conclusion, for all classes in the PROMISE-NFR dataset, the model achieved its best performance at 8 training epochs, beyond which a decline in F1-score was observed due to overfitting. This indicates that 8 epochs strike a balance between effective learning and generalization to unseen data. In contrast, for the larger FR_NFR dataset, the optimal performance was reached at 4 epochs, achieving an F1-score of 0.98 for the FR class and 0.96 for the NFR class, as shown in Table 4(b).

4.3.2. Effect of Batch Size Selection on BERT Fine-Tuning Performance

Batch size plays a crucial role in model convergence and generalization, especially when fine-tuning BERT for classification tasks. In our experiments on the PROMISE-NFR dataset, a batch size of 8 consistently yielded superior F1-scores across both FR and NFR classes compared to larger batch sizes of 16 and 32. As illustrated in Figure 5, these results suggest that smaller batch sizes are more effective for fine-tuning BERT on smaller datasets, likely due to better gradient estimation and reduced overfitting.

For the larger FR_NFR dataset, however, a batch size of 16 resulted in higher F1-scores for both classes, outperforming batch sizes of 8 and 32. These findings underscore the importance of aligning batch size selection with dataset scale. Specifically, smaller datasets benefit from smaller batch sizes, while larger datasets require moderately larger batch sizes to optimize learning efficiency and performance.

This observation is consistent with prior research, including (Devlin et al., 2019), which emphasizes the interdependence between dataset size and batch size in BERT fine-tuning. Selecting an appropriate batch size is therefore a critical step in optimizing model performance for SRC tasks.

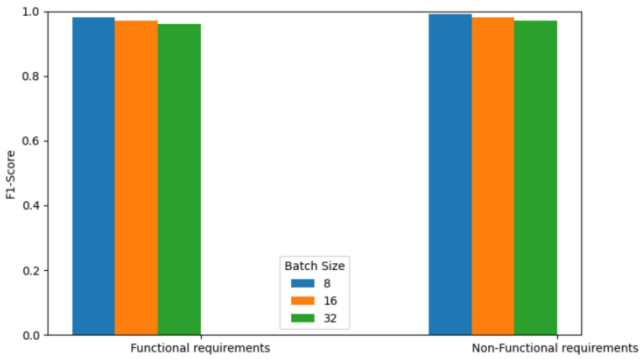


Figure 5. F1-Score across software requirement types and batch size settings in PROMISE NFR dataset.

4.3.3. Effect of Learning Rate on BERT Fine-Tuning Performance

 doi:10.20944/preprints202510.1480.v1

The learning rate is a critical hyperparameter in BERT fine-tuning, as it determines the step size for updating model parameters during training. Its selection directly affects both training stability and convergence speed, making it essential for achieving optimal performance (Taj et al., 2025).

In our experiments, we evaluated two learning rate settings: a lower rate of 1e-5 and a higher rate of 5e-5. As illustrated in Figure 6, the lower learning rate consistently yielded higher F1-scores across all requirement classes for both the PROMISE-NFR and FR_NFR datasets. These results suggest that a more conservative learning rate allows the model to fine-tune more precisely, especially on smaller datasets.

Interestingly, when both the batch size was increased to 32 and the learning rate to 5e-5 on the larger FR_NFR dataset, the model achieved performance comparable to the optimal configuration. This highlights the importance of dataset-size-aware hyperparameter tuning, where larger datasets can tolerate more aggressive learning rates and larger batch sizes without compromising performance.

These findings align with observations from the original BERT paper and the GLUE benchmark, which showed that different tasks and dataset sizes require tailored learning rate strategies to achieve optimal results.

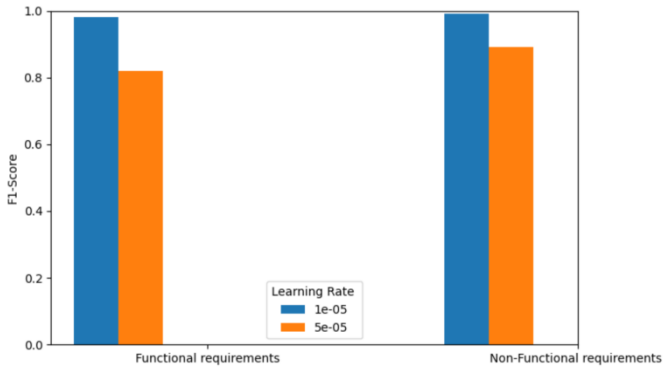


Figure 6. F1-Score across software requirement types and learning rates settings in PROMISE NFR dataset.

In summary, the hyperparameter configuration presented in Table 2 was derived from a thorough ablation study rather than relying on default settings. This process involved systematically testing various combinations to identify the most effective setup for optimizing model performance. For smaller datasets, the best results were achieved using a learning rate of 1e-5, batch size of 8, and 8 training epochs. For larger datasets, the optimal configuration included a learning rate of 1e-5, batch size of 16, and 4 training epochs. Across all experiments, we consistently applied the Adam optimizer, set the maximum sequence length to 256 tokens, and used a dropout rate of 0.2 to reduce overfitting. The classification head incorporated softmax activation and sparse categorical cross-entropy loss, enhancing the model's ability to distinguish between classes. Each hyperparameter was carefully selected based on empirical results, ensuring that the model was effectively tuned to learn task-specific patterns. These configurations reflect deliberate choices grounded in experimental evidence rather than arbitrary defaults.

5. Limitations and Future Work

This research lays a solid foundation for enhancing SRC through BERT fine-tuning. However, several limitations remain that should be addressed in future studies to further improve the model's applicability and performance.

One key limitation is the scale of dataset testing. While our model demonstrated strong performance on the PROMISE-NFR and FR_NFR datasets, it has not yet been evaluated on much larger software requirements datasets. Applying the model to datasets containing hundreds of thousands or even millions of requirements would require significantly more computational

 doi:10.20944/preprints202510.1480.v1

resources and extended training time. Such testing would help determine whether the current hyperparameter configurations—batch size 8 with 8 epochs for smaller datasets, and batch size 16 with 4 epochs for larger ones—remain effective at scale or need to be adjusted for larger collections.

Another limitation is the scope of classification. This study focused solely on binary classification between FRs and NFRs, without further distinguishing between specific NFR types. In practice, NFRs encompass a wide range of categories, including security, performance, usability, reliability, maintainability, and portability. Future work should aim to develop models capable of multi-class classification to identify and categorize these distinct NFR types.

To address these limitations, future research should focus on three main directions:

1. Scaling the model to handle larger software requirements datasets while maintaining efficiency and accuracy.
2. Developing specialized models for fine-grained classification of individual NFR types.
3. Creating adaptive tuning mechanisms that automatically adjust hyperparameters based on dataset characteristics.

These enhancements will contribute to building more robust, scalable, and intelligent SRC systems capable of supporting real-world software engineering tasks.

6. Conclusions

This study presented a BERT-based approach for the SRC task, with a strong emphasis on systematic hyperparameter optimization. The primary goal was to automate the classification of FRs and NFRs by leveraging the capabilities of advanced natural language processing techniques. Through careful fine-tuning of the BERT model, we significantly enhanced its performance by optimizing key hyperparameters such as batch size, learning rate, and the number of training epochs.

Experiments were conducted on two datasets of varying sizes: the PROMISE-NFR dataset and the larger FR_NFR dataset. Our proposed fine-tuning methodology consistently outperformed existing baseline models, delivering high accuracy and reliable classification results. The ablation study allowed us to identify optimal configurations tailored to dataset size: a batch size of 8 with 8 training epochs for smaller datasets, and a batch size of 16 with 4 training epochs for larger datasets. Both configurations used a learning rate of 1e-5 and the Adam optimizer, ensuring stable and effective training.

Based on the experimental results, three key conclusions can be drawn. First, BERT fine-tuning with optimized hyperparameters significantly outperforms traditional machine learning models such as SVM, Naïve Bayes, and logistic regression in SRC tasks. Second, systematic hyperparameter tuning is essential for achieving optimal performance, especially for smaller datasets where smaller batch sizes tend to yield better results. Third, the relationship between dataset size, batch size, learning rate, and training epochs is critical to successful BERT fine-tuning. Our findings are consistent with established best practices in transformer-based model optimization.

Overall, this study demonstrates that BERT fine-tuning, when properly optimized, is a powerful and scalable solution for automated software requirements classification. It offers strong potential for practical applications in software engineering, particularly in improving the efficiency and accuracy of requirements analysis.

Author Contributions: Safaa Eltahier designed the study and wrote the first draft of the manuscript. Omer Dawood and Imtithal Saeed contributed to implementation and reviewing the manuscript. All authors provided funding and read and approved the final version of the manuscript.

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

References

Abad, Z.S.H., 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).

 doi:10.20944/preprints202510.1480.v1

Presented at the 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 496–501. https://doi.org/10.1109/RE.2017.36

Airlangga, G., 2024. Enhancing Software Requirements Classification with Semisupervised GAN-BERT Technique. J. Electr. Comput. Eng. 2024, 4955691. https://doi.org/10.1155/2024/4955691

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). Presented at the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), pp. 610–615. https://doi.org/10.1109/COMPSAC.2019.10275

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: Burstein, J., Doran, C., Solorio, T. (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Presented at the NAACL-HLT 2019, Association for Computational Linguistics, Minneapolis, Minnesota, pp. 4171–4186. https://doi.org/10.18653/v1/N19-1423

EzzatiKarami, M., Madhavji, N.H., 2021. Automatically Classifying Non-functional Requirements with Feature Extraction and Supervised Machine Learning Techniques: A Research Preview, in: Dalpiaz, F., Spoletini, P. (Eds.), Requirements Engineering: Foundation for Software Quality. Springer International Publishing, Cham, pp. 71–78. https://doi.org/10.1007/978-3-030-73128-1_5

Gardazi, N.M., Daud, A., Malik, M.K., Bukhari, A., Alsahfi, T., Alshemaimri, B., 2025. BERT applications in natural language processing: a review. Artif. Intell. Rev. 58, 166. https://doi.org/10.1007/s10462-025-11162-5

Gkouti, N., Malakasiotis, P., Toumpis, S., Androutsopoulos, I., 2024. Should I try multiple optimizers when fine-tuning pre-trained Transformers for NLP tasks? Should I tune their hyperparameters? https://doi.org/10.48550/arXiv.2402.06948

Haque, Md.A., Abdur Rahman, Md., Siddik, M.S., 2019. Non-Functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study, in: 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT). Presented at the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), pp. 1–5. https://doi.org/10.1109/ICASERT.2019.8934499

Hey, T., Keim, J., Koziolek, A., Tichy, W.F., 2020a. NoRBERT: Transfer Learning for Requirements Classification, in: 2020 IEEE 28th International Requirements Engineering Conference (RE). Presented at the 2020 IEEE 28th International Requirements Engineering Conference (RE), IEEE, Zurich, Switzerland, pp. 169–179. https://doi.org/10.1109/re48521.2020.00028

Hey, T., Keim, J., Koziolek, A., Tichy, W.F., 2020b. NoRBERT: Transfer Learning for Requirements Classification, in: 2020 IEEE 28th International Requirements Engineering Conference (RE). Presented at the 2020 IEEE 28th International Requirements Engineering Conference (RE), IEEE, Zurich, Switzerland, pp. 169–179. https://doi.org/10.1109/re48521.2020.00028

Jindal, R., Malhotra, R., Jain, A., Bansal, A., 2021. Mining Non-Functional Requirements using Machine Learning Techniques. E-Inform. Softw. Eng. J. 15. https://doi.org/10.37190/e-inf210105

Kaur, K., Kaur, P., 2024. The application of AI techniques in requirements classification: a systematic mapping. Artif. Intell. Rev. 57. https://doi.org/10.1007/s10462-023-10667-1

Kaur, K., Kaur, P., 2023. BERT-CNN: Improving BERT for Requirements Classification using CNN. Procedia Comput. Sci. 218, 2604–2611. https://doi.org/10.1016/j.procs.2023.01.234

Kici, D., Malik, G., Cevik, M., Parikh, D., Başar, A., 2021. A BERT-based transfer learning approach to text classification on software requirements specifications. Proc. Can. Conf. Artif. Intell. https://doi.org/10.21428/594757db.a4880a62

Kingma, D.P., Ba, J., 2017. Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/arXiv.1412.6980

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). Presented at the 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 490–495. https://doi.org/10.1109/RE.2017.82

Li, G., Zheng, C., Li, M., Wang, H., 2022. Automatic Requirements Classification Based on Graph Attention Network. IEEE Access 10, 30080–30090. https://doi.org/10.1109/ACCESS.2022.3159238

Luo, X., Xue, Y., Xing, Z., Sun, J., 2022. PRCBERT: Prompt Learning for Requirement Classification using BERT-based Pretrained Language Models, in: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. Presented at the ASE '22: 37th IEEE/ACM International Conference on Automated Software Engineering, ACM, Rochester MI USA, pp. 1–13. https://doi.org/10.1145/3551349.3560417

Navarro-Almanza, R., Juarez-Ramirez, R., Licea, G., 2017. Towards Supporting Software Engineering Using Deep Learning: A Case of Software Requirements Classification, in: 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT). Presented at the 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT), pp. 116–120. https://doi.org/10.1109/CONISOFT.2017.00021

PROMISE NFR, 2005. https://github.com/AleksandarMitrevski/se-requirements-classification/blob/master/0-datasets/PROMISE_exp/PROMISE_exp.arff

PURE: A Dataset of Public Requirements Documents [WWW Document], 2017. URL https://ieeexplore.ieee.org/document/8049173 (accessed 9.16.24).

Shreda, Q.A., Hanani, A.A., 2025. Identifying Non-Functional Requirements From Unconstrained Documents Using Natural Language Processing and Machine Learning Approaches. IEEE Access 13, 124159–124179. https://doi.org/10.1109/access.2021.3052921

Sommerville, I., Sawyer, P., 1999. Requirements Engineering: A Good Practice Guide. Wiley, Chichester, Eng. ; New York.

Sonali, S., Thamada, S., 2024. FR_NFR_dataset. https://doi.org/10.17632/4ysx9fyzv4.

Taj, S., Daudpota, S.M., Imran, A.S., Kastrati, Z., 2025. Aspect-based sentiment analysis for software requirements elicitation using fine-tuned Bidirectional Encoder Representations from Transformers and Explainable Artificial Intelligence. Eng. Appl. Artif. Intell. 151, 110632. https://doi.org/10.1016/j.engappai.2025.110632

Wiegers, K.E., Beatty, J., Wiegers, K.E., 2013. Software requirements, 3. ed. [fully updated and expanded]. ed, Best practices. Microsoft Press, Redmond, Wash.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Platen, P. von, Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M., 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. https://doi.org/10.48550/arXiv.1910.03771