A Study to Evaluate the Impact of LoRA Fine-tuning on the Performance of Non-functional Requirements Classification

Xia Li, Allen Kim

The Department of Software Engineering and Game Design and Development,
Kennesaw State University,
Marietta, USA

Abstract. Classifying Non-Functional Requirements (NFRs) in software development life cycle is critical. Inspired by the theory of transfer learning, researchers apply powerful pre-trained models for NFR classification. However, full fine-tuning by updating all parameters of the pre-trained models is often impractical due to the huge number of parameters involved (e.g., 175 billion trainable parameters in GPT-3). In this paper, we apply Low-Rank Adaptation (LoRA) fine-tuning approach into NFR classification based on prompt-based learning to investigate its impact. The experiments show that LoRA can significantly reduce the execution cost (up to 68% reduction) without too much loss of effectiveness in classification (only 2%-3% decrease). The results show that LoRA can be practical in more complicated classification cases with larger dataset and pre-trained models.

Keywords: Non-functional requirements classification, low-rank adaptation (LoRA), pre-trained models, fine-tuning

1 Introduction

Non-Functional Requirements (NFRs) in the whole software development life cycle are critical for meeting users' expectations and ensuring the system performs well across various dimensions, such as quality, usability, security, and performance. Unlike functional requirements that focus on the specific actions of the system, NFRs define the overall behavior and constraints of the system, which makes them essential in guiding architectural design decisions. For example, NFRs help determine how secure, scalable, or reliable the system needs to be. Given their importance, it is essential to identify and extract NFRs from software requirements specification (SRS) documents early in the development process to ensure that they are adequately addressed. However, categorizing NFRs is a time-consuming task due to the complexity of their descriptions with natural languages. Consequently, the process is error-prone and requires a significant effort to ensure that all NFRs are properly understood and incorporated into the system design.

David C. Wyld et al. (Eds): SIPO, BDHI, NET, SOEA, CSML, AISCA, MLIOB, DNLP – 2025
pp. 103-111, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.150410

In the past decades, various machine learning and deep learning techniques have been applied to classify Non-Functional Requirements. For example, EzzatiKarami et al. [3] used Support Vector Machine (SVM) and Decision Tree for NFR classification by combining different feature extraction techniques. Rahimi et al. [9] used four deep learning models (e.g., LSTM, BiLSTM, GRU, and CNN) for accurate requirements classification. With the development and application of transformer models [22], various powerful pre-trained models (e.g., BERT [4], GPT [5]) have been applied into requirements classification inspired by the theory of transfer learning. The basic idea is that the pre-trained foundation models can be trained in advance and users can fine-tune the models for their specific downstream tasks. For example, Hey et al. [10] proposed NoRBERT to fine-tune the BERT model and apply it to different tasks for requirements classification. Luo et al. [7] proposed PRCBERT by applying prompt engineering for requirement classification using BERT model with flexible prompt templates. Despite the promising performances of current studies with pre-trained models, the major issues is the cost to fine-tune the models. Currently, there are huge amount of parameters (up to billions) to be tuned for modern large language models, indicating that a lot of time and resources will be spent to fine-tune better results for the specific task. To overcome the problem, a parameter-efficient fine-tuning (PEFT) approach called Low-Rank Adaptation (LoRA [12]) is proposed to accelerate the fine-tuning of large models while consuming less memory. In this paper, we propose to conduct an extensive study to evaluate the impact of LoRA on NFR classification. We apply p-tuning [6] which is a prompt-based approach on various pre-trained models such as BERT and RoBERTa [23]. The reason to use p-tuning is that it can provide stable performance for classification tasks by designing learnable templates. Our study indicates that LoRA can significantly reduce the execution cost (e.g., up to 68% reduction) without too much decrease of classification abilities.

The structure of the paper is as follows. In Section 2, we introduce the studies related to software requirements classification. In Section 3, we illustrate the approach of our study. In Section 4 and Section 5, we discuss our experimental configurations as well as the results analysis, respectively. We discuss the threats to validity in Section 6 and conclude the paper in Section 7.

2 Related Work

2.1 Traditional learning techniques for requirements classification

In recent years, machine learning (ML) and deep learning (DL) techniques have been increasingly applied to enhance software requirement classification. For example, Canedo et al. [13] compare the performance of different models such as Logist Regression (LR), Support Vector Machine (SVM), Multinomial Naive Bayes (MNB)

and K-Nearest Neighbors (KNN) for both functional and non-functional requirements. Amasaki et al. [2] use vectorization methods (e.g., document embedding methods) with various supervised machine learning models for NFR Classification. AlDhafer et al. [14] use Bidirectional Gated Recurrent Neural Networks (BiGRU) to classify requirements by considering word sequences and character sequences as tokens. Dekhtyar et al. [8] use Word2Vec embeddings and CNN for NFR classification.

2.2 Pre-trained models for requirements classification

Inspired by the theory of transfer learning, researchers seek to apply powerful pre-trained models into the filed of software requirements classification. Kici et al. [15] fine-tune an existing pre-trained model called DistilBERT to achieve promising performance compared with other deep learning methods such as LSTM and BiLSTM. Hey et al. [9] propose a new approach called NoRBERT to fine-tune the popular BERT model for requirements classification. Luo et al. [7] apply prompt learning for requirements classification by using BERT model flexible prompt templates to achieve accurate requirements classification. Kaur et al. [16] present a new Bidirectional Encoder-Decoder Transformer-Convolutional Neural Network (BERT-CNN) model for requirements classification, performing better than the state-of-the-art baseline approach.

2.3 LoRA for classification tasks

LoRA (Low-Rank Adaptation) fine-tuning is a technique used to efficiently adapt pre-trained large language models (LLMs) to specific tasks with minimal computational cost. Researchers have applied LoRA into various fields for classification tasks to achieve promising performance. Yang et al. [17] introduce an advanced methodology for financial news topic classification by leveraging the Chatglm3-6b model [18] through LoRA and Noise Enhanced Fine-Tuning (NEFT). Shuai et al. [19] introduces a method utilizing the Llama3-8b model for emotion text classification which is accelerated by LoRA and FlashAttention techniques. McCleary et al. [20] explore the application of LoRA fine-tuning approach with small language models for triple negative breast cancer cases with various large language models such as GPT 3.5, GPT 4 and Mistral's 7B. Aggarwal et al. [21] apply LoRA to the realm of image classification for plant disease detection.

In this paper, we conduct an extensive study on the impact of LoRA by fine-tuning various large language models such as BERT and RoBERTa. We apply p-tuning, which is a popular prompt-based learning technique through pre-trained models by designing learnable templates fed into the models for training.

3 Study Approach

In this section, we introduce how we conduct the experiments for NFR classification based on p-tuning and LoRA fine-tuning. The overall process is shown in Figure 1, including following steps. Please note that we cite the famous illustration of LoRA [12] as the fine-tuning part in the figure. First, we convert the original requirement statements based on a specific template with learnable tokens based on p-tuning. Second, the new template is fed into different pre-trained models to predict the category of requirement. Third, during the training process, we apply LoRA approach to freeze the weights of original pre-trained model by adding an alternative matrices. The target label can be predicted through back propagation with updated weights. Next, we introduce the basic ideas of p-tuning and LoRA used in our study.

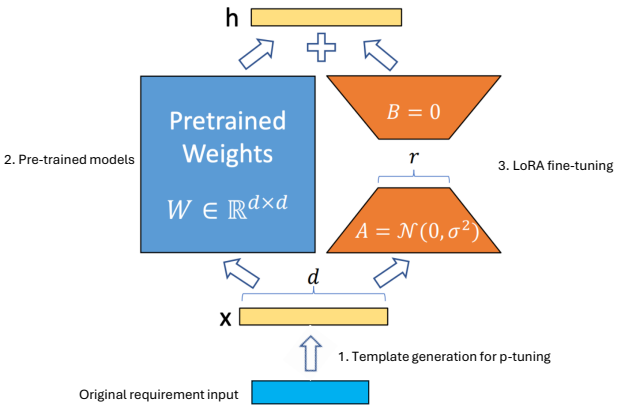

Fig. 1. Overview of the study approach

3.1 P-tuning

P-tuning is a prompt-based learning technique where certain learnable continuous prompt embeddings (soft templates) are created in concatenation with original input requirement sequence. The target label of the classification task is embedded in the sequence that pre-trained models can predict. For other techniques using pre-trained models, an additional neural network (e.g., RNN, CNN) is connected with the pre-trained models and pre-trained models are typically used to generate a final vector representation for input sequence with low correlation between the input sequence and the target task. However, p-tuning can let pre-trained models directly predict the masked token label to achieve better classification performance.

For p-tuning, we use the following template as the input of the pre-trained models: *[CLS][P][P][P]REQUIEMRNT SENTENCE[SEP]. [CLS][P][P][P]This requirement is related to [M][SEP].* In the template design, [P] represents the learnable token without any real meanings while [M] is the masked token to represent the requirement category (e.g., performance, security, usability) that can be predicted by pre-trained models. [CLS] represents a special token in the front of the original input text and [SEP] is a separator token to represent the segment of each sentence. Thus, one example of the final input can be " *[CLS][P][P][P] How well are the system and its data protected against attacks?[SEP]. [CLS][P][P][P]This requirement is related to [M][SEP]* ".

3.2 LoRA

To adapt current pre-trained large language models (e.g., GPT-3, LLama, and Mistral) to specific tasks, fine-tuning is an essential step. However, full fine-tuning by updating all parameters of the pre-trained models is often impractical due to the huge number of parameters involved (e.g., 175 billion trainable parameters in GPT-3), making it computationally expensive and storage-intensive. To resolve this problem, LoRA is proposed to offer a more efficient alternative by significantly reducing the number of trainable parameters. As shown in Figure 1, LoRA can freeze the weights of pre-trained models while insert two smaller matrices to represent the updated weights through low-rank decomposition. These new matrices can be trained to adapt to the data while keeping the overall number of changes low. In detail, W0 with size $d \times d$ is the original weight matrix and kept unchanged during the training procedure. ΔW is the new updated weights with the same size $d \times d$. However, in LoRA, a new parameter r is introduced to reduce the size of the matrix ΔW. The smaller matrices, A and B, are defined with a reduced size of $r \times d$, and $d \times r$. For LoRA, only the weights of matrices A and B are trained to update during training process. After the training process is completed, the new input x with a size of $1 \times d$ will be multiplied by both W and ΔW, resulting in two d-sized output vectors. These vectors are then concatenated element-wise to produce the final result, and the weights can be merged with the base model. The final output is a vector with size h shown in the figure. In this paper, we investigate how LoRA can improve the efficiency of NFR classification and if the similar effectiveness can be kept with LoRA.

4 Experimental Design

In this section, we introduce the pre-trained models and dataset used in our study in Section 4.1 as well as the experimental configuration and evaluation metrics in Section 4.2.

4.1 Pre-trained models and dataset

To evaluate LoRA on pre-trained models with different sizes, we use four foundation models "bert-base-uncased" (110M parameters), "bert-large-uncased" (340M parameters), "roberta-base"(125M parameters) and "roberta-large"(355M parameters) that can be downloaded from the popular AI community Hugging Face[1]. We use the widely used dataset PROMISE [1] with 914 non-functional requirements that consists of the following five pre-labeled categories: operability, maintainability, performance, security, and usability. To make the dataset adapted to the template, we apply popular natural language processing steps (e.g., stemming, lemmatization, stop-word removal and conversion to lower case) via the widely used NLTK[2] toolkit to pre-process the dataset. We also remove special characters that are unique in different domains.

4.2 Experimental configuration and evaluation metrics

In our study, we split the original dataset into training set (80%) and test set (20%). We apply 10-fold cross-validation for the four pre-trained models. We use cross-entropy loss function since NFR classification is the classic multi-class classification problem. We also use the popular evaluation metrics precision, recall and F1 score for classification problems. Their equations are as follows.

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where TP represents the number of True Positives, FP indicates the number of False Positives and FN is the number of False Negatives.

For the hyperparameters, we set the maximum input sequence length as 128, batch size as 8, learning rate as $3e^{-5}$, epochs as 32. To implement LoRA, we use the pre-defined framework **peft** from Hugging Face. From the experiments, we can reduce the trained parameters to 20% - 25% from original pre-trained models. Also, we set the specific parameter rank r used in LoRA as 8. We also use AdamW optimizer [11] in the training process. All training and inference are executed on a server with Intel Core 13900K CPU, 32GB memory and NVIDIA RTX 4090 GPU.

5 Results Analysis

In this section, we illustrate the impact of LoRA on the efficiency and effectiveness of NFR classification with p-tuning, shown in Table 1 and Table 2. In Table 1,

[1] Hugging Face. https://huggingface.co/
[2] NLTK toolkit. https://www.nltk.org/

Table 1. Efficiency of NFR classification with and without LoRA

Model	without LoRA	with LoRA	%reduction
Bert-base	11 mins	5 mins	54.5%
Bert-large	19 mins	6 mins	68.4%
Roberta-base	13 mins	7 mins	46.2%
Roberta-large	20 mins	9 mins	55%

Table 2. Effectiveness of NFR classification with and without LoRA

Model	Precision	Recall	F1 score
Bert-base	81.39%(78.75%)	82.46%(78.13%)	81.92%(78.40%)
Bert-large	82.37%(79.26%)	82.90%(79.38%)	82.44%(79.32%)
Roberta-base	82.57%(80.73%)	83.17%(79.75%)	82.86%(80.26%)
Roberta-large	83.45%(81.29%)	83.57%(80.15%)	83.42%(80.78%)

the number indicates the execution cost of classification for 1-fold cross validation. From the results, we can find that LoRA can significantly reduce the execution cost of NFR classification. For example, for the Besrt-large model, it can reduce the cost by 68.4% (19 mins vs 6 mins). In Table 2, the first column represents the four models used in the study. The last three columns represent the values in terms of the three metrics: precision, recall and F1 score. Please note that the number outside the parentheses represents the value without LoRA while the number in the parentheses indicates corresponding results with LoRA fine-tuning. Also, all results are calculated as the average values of 10-fold cross-validation. From the results, we have following findings. First, we can find that the two models from Roberta performs better than Bert models. For example, the F1 score of Bert-base is 81.92% while it is 82.86% for RoBERTa. The possible reasons could be that RoBERTa is trained on a larger corpus of text with robust representation of the language and it uses a dynamic masking strategy where different tokens are masked in each training example. Also, we can find that the large versions of both models perform better than their corresponding base model. It indicates that more weights trained for NFR classification can get better performance. Third, with LoRA fine-tuning approach, we can find that the performance are worse than corresponding models without LoRA. However, the difference is only 2%-3%, showing that LoRA can be practical and acceptable in more complicated classification cases with larger dataset and pre-trained models.

6 Threats to Validity

The main external threat to the validity is the dataset we used. In our study, we use the widely used data PROMISE for NFR classification to make our results as

accurate as possible. The internal threat to the validity is the implementation of the training and inference. To reduce the threat, we use the pre-defined framework of LoRA from the popular AI community Hugging Face.

7 Conclusion

In this paper, we conducted a comprehensive study to evaluate impact of LoRA fine-tuning approach on the performance of prompt-based non-functional requirements classification. Our experimental results show that with LoRA fine-tuning approach, the execution cost can be reduced by up to 68% and the performance are worse than corresponding models without LoRA. However, the difference is only 2%-3%, showing that LoRA can be practical and acceptable in more complicated classification cases with larger dataset and pre-trained models compared with the significant execution cost reduction. In future, we plan to evaluate NFRs classification by using more fine-tuning approaches.

References

1. Sayyad, SJ. PROMISE software engineering repository. 2005
2. Amasaki, Sousuke and Leelaprute, Pattara. The effects of vectorization methods on non-functional requirements classification. 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)
3. EzzatiKarami, Mahtab and Madhavji, Nazim H, Automatically classifying non-functional requirements with feature extraction and supervised machine learning techniques: A research preview. Requirements Engineering: Foundation for Software Quality: 27th International Working Conference, REFSQ 2021, Essen, Germany, April 12–15, 2021.
4. Devlin, Jacob. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
5. Brown, Tom B. Language models are few-shot learners. arXiv preprint arXiv:2005.14165. 2020
6. Liu, Xiao and Zheng, Yanan and Du, Zhengxiao and Ding, Ming and Qian, Yujie and Yang, Zhilin and Tang, Jie. GPT understands, too. AI Open, 2023
7. Luo, Xianchang and Xue, Yinxing and Xing, Zhenchang and Sun, Jiamou. Prcbert: Prompt learning for requirement classification using bert-based pretrained language models. Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, 2022
8. Dekhtyar, Alex and Fong, Vivian. Re data challenge: Requirements identification with word2vec and tensorflow. 2017 IEEE 25th International Requirements Engineering Conference (RE)
9. Rahimi, Nouf and Eassa, Fathy and Elrefaei, Lamiaa. One-and two-phase software requirement classification using ensemble deep learning. Entropy, 2021
10. Hey, Tobias and Keim, Jan and Koziolek, Anne and Tichy, Walter F. Norbert: Transfer learning for requirements classification. 2020 IEEE 28th international requirements engineering conference (RE)
11. Loshchilov, I. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101. 2017
12. Hu, Edward J and Shen, Yelong and Wallis, Phillip and Allen-Zhu, Zeyuan and Li, Yuanzhi and Wang, Shean and Wang, Lu and Chen, Weizhu. Lora: Low-rank adaptation of large language models, arXiv preprint arXiv:2106.09685, 2021.
13. Dias Canedo, Edna and Cordeiro Mendes, Bruno, Software requirements classification using machine learning algorithms.Entropy, 2020, MDPI

14. AlDhafer, Osamah and Ahmad, Irfan and Mahmood, Sajjad. An end-to-end deep learning system for requirements classification using recurrent neural networks. Information and Software Technology. 2022
15. Kici, Derya and Malik, Garima and Cevik, Mucahit and Parikh, Devang and Basar, Ayse. A BERT-based transfer learning approach to text classification on software requirements specifications. Canadian AI. 2021.
16. Kamaljit Kaur, Parminder Kaur. BERT-CNN: Improving BERT for Requirements Classification using CNN. Procedia Computer Science. 2023
17. Yang, Liziqiu and Huang, Yanhao and Tan, Cong and Wang, Sen. News topic classification base on fine-tuning of chatglm3-6b using neftune and lora. Proceedings of the 2024 International Conference on Computer and Multimedia Technology.2024
18. Du, Zhengxiao and Qian, Yujie and Liu, Xiao and Ding, Ming and Qiu, Jiezhong and Yang, Zhilin and Tang, Jie. Glm: General language model pretraining with autoregressive blank infilling. arXiv preprint arXiv:2103.10360. 2021
19. Shui, Hongyi and Zhu, Yuanjing and Zhuo, Fan and Sun, Yibo and Li, Daoyuan. An Emotion Text Classification Model Based on Llama3-8b Using Lora Technique. 2024 7th International Conference on Computer Information Science and Application Technology (CISAT). 2024
20. McCleary, Kyle and Ghawaly, James and Miele, Lucio. TNM Tumor Classification from Unstructured Breast Cancer Pathology Reports using LoRA Finetuning of Mistral 7B. AAAI 2024 Spring Symposium on Clinical Foundation Models. 2024
21. Aggarwal, Deeksha and Mittal, Yash and Kumar, Uttam. Advancing Image Classification through Parameter-Efficient Fine-Tuning: A Study on LoRA with Plant Disease Detection Datasets. The Second Tiny Papers Track at ICLR.2024
22. Vaswani, A. Attention is all you need. Advances in Neural Information Processing Systems. 2017
23. Liu, Yinhan. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692. 2019