Improving Requirements Classification Models based on Explainable Requirements Concerns

1st Lu Han
Faculty of Information Technology
Beijing University of Technology
Beijing, China
hanlu1997@emails.bjut.edu.cn

2nd QiXiang Zhou
Faculty of Information Technology
Beijing University of Technology
Beijing, China
qixiang.zho@gmail.com

3rd Tong Li*
Faculty of Information Technology
Beijing University of Technology
Beijing, China
litong@bjut.edu.cn
*corresponding author

***Abstract*—Requirements classification is an important step in requirements analysis. Recent studies mostly adopt machine learning techniques to automate requirements classification. In particular, state-of-the-art technologies such as Bidirectional Encoder Representations from Transformers (BERT) have significantly improved the accuracy of classification tasks. However, most of these models are considered black-box models, and thus the rationale of the classification is unclear, affecting the improvement of the classification model. In this paper, we propose a technique for improving requirements classification models based on an explainable AI (XAI) framework. Specifically, our proposed approach first trains a concern extraction model to identify requirements concerns. Then, the requirements classification model is analyzed using an explainability framework to generate the explanation, which may shed light on the noise in the dataset. Finally, we denoise the training dataset and fine-tune the model to improve its performance. We evaluate our proposed technique using an existing requirements classification approach and two existing requirements datasets. The experimental results demonstrate that our approach significantly improves the model performance. Specifically, the accuracy improved by 7.68%, the recall improved by 7.44%, and the F1 improved by 7.80% overall.**

***Index Terms*—requirements classification, requirements concerns, model evolution, explainability**

I. Introduction

Requirements classification is a crucial step in requirements analysis, which involves categorizing different requirements based on their characteristics and attributes. This process helps identify the critical features of the requirements and determine how to address them effectively. In recent years, machine learning techniques have been adopted to automate the requirements classification process. These techniques have significantly improved the accuracy of classification tasks by providing efficient and reliable methods for analyzing large volumes of requirements data. The use of BERT in the classification of requirements has proved highly effective in recent studies. Researchers have reported significant improvements in accuracy when using BERT compared to traditional machine learning models. BERT has also proven to be more robust in handling noisy and unstructured data, making it a valuable asset for requirements classification in real-world scenarios. Machine learning techniques, especially BERT, have revolutionized requirements classification by providing efficient and accurate methods for analyzing large volumes of requirements data. As the demand for high-quality software products continues to increase, the need for automated requirements classification will become more critical, and machine learning techniques like BERT will undoubtedly play a significant role in meeting this demand.

Although machine learning techniques such as BERT have significantly improved the accuracy of requirements classification, they still pose significant challenges. One major challenge is that most of these models are considered to be black-box models. Humans often find their internal reasoning mechanisms difficult to interpret and understand. As a result, it can be challenging to identify the factors contributing to the model's low accuracy or to troubleshoot issues when problems arise. Another challenge is that the factors contributing to the model's low accuracy can be ambiguous or unclear. In some cases, the model may need help to capture critical features of specific requirements due to differences in language use or other nuances. The quality of the training data used to train the model can also affect its accuracy. Low-quality or noisy data can lead to errors and inconsistencies in the model output.

We propose a novel technique based on an explainable AI (XAI) framework to address the challenges associated with requirements classification models. We first train a concerns extraction model to identify requirements' concerns and then analyze the model using an explainability framework to understand its internal reasoning mechanisms better. This analysis enables us to identify areas where the model may be struggling and to make improvements to improve its performance. Then we denoise the original dataset to remove any inconsistencies or errors that may impact the model's accuracy. This step is critical in ensuring that the model is trained on high-quality data, which can significantly impact its overall performance. Finally, we fine-tune the original model using the denoised dataset to further improve its accuracy and performance.

To validate the effectiveness of our proposed technique, we conducted experiments using existing requirements classification approaches. Specifically, we evaluate the performance of our approach on a widely used BERT model. The results of our experiments showed that our proposed technique significantly improved the performance of the requirements classification model. Overall, the results of our experiments demonstrate that our proposed technique offers a promising solution to the challenges associated with requirements classification models. By incorporating XAI techniques and optimizing the model training process, we significantly improved the model's performance and provided more transparent and interpretable insights into its decision-making process.

In the rest of the paper, we present related work in Section II. In Section III, we present the technique of debug requirements and classification model. In Section IV, we present an experiment to evaluate the approach and offer the evaluation results. In section V, we present a discussion of the approach. Finally, we give the conclusions in Section VI.

II. Related Work

Requirements classification is critical to developing software systems that meet stakeholder expectations. A significant challenge in requirements engineering is the classification of functional requirements and non-functional requirements. Machine learning (ML) techniques have been used to automate requirements classification, and several studies have been conducted in this area. Binkhonain and Zhao [1] thoroughly reviewed various ML algorithms to identify and classify requirements. Similarly, Lu and Liang [2] proposed an approach that uses augmented user application reviews to classify requirements. Hey et al. [3] presented NorBERT, a transfer learning-based method that leverages pre-trained language models to classify requirements accurately. Finally, Winkler and Vogelsang [4] proposed a convolutional neural network-based approach for automatically classifying requirements. In addition to the automatic classification of functional requirements and non-functional requirements, some scholars have proposed a new requirements classification system. Glinz [5] proposed a concern-based taxonomy of requirements. Concerns can be functional or behavioral, performance-related, or quality-related, with each type addressing specific aspects of the system.

In recent years, machine learning techniques have brought a lot of convenience to people's lives. More and more people are using machine learning to requirements classification tasks. But due to their black-box nature, the logic inside the models is wholly hidden from developers. Even experts cannot fully understand the reason behind this analysis [6]. Thus, the requirement for explainability has arisen. Explainability is a critical functional requirement for AI systems due to the need for transparency, trust, and alignment with user expectations and objectives. Explainability helps identify and mitigate potential bias in machine learning models or helps develop and improve machine learning models [7]. Ribeiro proposed LIME [8], a novel explanation technique that explains the predictions of any classifier in an interpretable and faithful manner and a non-redundant method for explaining models through individual representative predictions. It improves trust, identifies model deficiencies, and provides insights into different text and image classification models. The authors present a novel unified framework, SHAP, for interpreting the predictions of complex models that assign each feature an importance value for a particular prediction and unifies six existing methods into a new class with unique properties to improve computational performance and consistency with human intuition. Lundberg [9] presented a novel unified framework, SHAP, for interpreting the predictions of complex models that assign each feature an importance value for a particular prediction and unifies six existing methods into a new class with unique properties to improve computational performance and consistency with human intuition. Ezzini, Saad and Sallam et al. [10] developed TAPHSIR. This tool addresses anaphoric ambiguity in requirements by detecting and resolving misunderstandings caused by pronouns during development. Dalpiaz and Dell'Anna et al. [11] propose to use linguistic features for constructing interpretable machine learning classifiers in requirements engineering. This approach improves interpretability and performs better on validation datasets. The benefits of explainability are clear, and the importance of this requirement is increasingly recognized by both industry and government [12].

Another critical area in natural language processing (NLP) is debugging of NLP models. More and more people choose to use explanation to assist model debugging. Users give feedback after receiving the explanation and debug the model according to the feedback to improve the model's performance. Lertvittayakumjorn and Toni [13] provided an overview of explanation-based human debugging of NLP models, highlighting the importance of human explanations for improving the model's performance. Similarly, Hartmann and Sonntag [14] presented a survey on techniques for improving NLP models using human explanations. On the other hand, different debugging methods and approaches have been proposed, such as influence functions (Koh and Liang [15]), HILDIF (Zylberajch et al., [16]), and FIND (Lertvittayakumjorn et al., [17]), that allow human-in-the-loop debugging of deep text classifiers.

Overall, automatic requirements classification and explainability techniques are increasingly important in software engineering, improving system transparency and trust, and reducing bias, with human-in-the-loop debugging methods also proving effective in improving NLP models.

III. Model Evolution Based On Explainability

In previous text analysis work, methods such as Naive Bayes, SVM, and K-NN were frequently used due to their simplicity in the reasoning process. Naive Bayes is a probabilistic Machine Learning technique that relies on Bayes's theorem. SVM is a supervised Machine Learning algorithm that applies a non-probabilistic approach to solve a classification problem. At the same time, K-NN is an instance-based learning algorithm that determines the class of an unknown sample by comparing it to the K-nearest-neighbors to it.

However, more precise methods have emerged as deep learning progresses, including LSTM and transformers. These algorithms can capture complex and long-term dependencies in sequences, making them effective in tasks such as text generation, sentiment analysis, and machine translation.

Although these advanced techniques are used to understand the text more precisely, they often come at the cost of interpretability. Black-box models like deep neural networks lack clear transparency and reasoning ability, which makes it challenging to understand how they arrive at their predictions. As the most important part of model development, the method of model debugging is usually decisive. Standard model debugging methods include optimizing model performance using cross-validation, adjusting parameters and structure, increasing training data sets, etc. They can effectively improve the model's predictive ability and generalization ability. However, traditional model debugging methods do not have a deep understanding of the reasoning process of the model, making it difficult to interpret.

To better understand the process of model predictions, interpretability methods can express how the model makes predictions in a human-readable way. These approaches include using local interpretability techniques such as LIME, SHAP, etc., to explain the cause and effect of individual predictions and global interpretability techniques such as neural network visualization, rule extraction, etc., to gain a thorough understanding of the entire model's behavior and prediction logic. These methods can help us better understand the model's behavior, strengthen trust in the model, and can be used to find holes in the model and improve the setting. Therefore, understanding and using interpretability methods is important to model development, improving models' reliability and predictive power.

We proposed a framework to debug the requirement classification model. The workflow is shown below.1

A. Extract requirements concerns

We train a concern extraction model to find which words are decisive for requirement classification (i.e., concerns). Specifically, we annotated the PROMISE NFR dataset and employed BERT to train our model. As a result, the model can receive a requirement text as input and generate a set of concerns associated with the text as output.

1) Dataset: To facilitate our annotation process, we leveraged the PROMISE NFR dataset [18], a well-established and extensively employed resource within the research community. This dataset encompasses 625 examples, serving as a valuable benchmark for various studies in the field. Within this dataset, there is a diverse range of 625 requirements, each of which can be classified as functional or non-functional. Specifically, 255 requirements are functional, representing the intended behaviors and actions of the system. On the other hand, the dataset also includes 370 non-functional requirements, which capture the system's desired qualities, constraints, and performance aspects. For accessibility, the dataset is readily available at the following link[1], allowing researchers and practitioners alike to explore and utilize it for their investigations.

Building upon the comprehensive concern-based taxonomy of requirements proposed by Glinz [5], Hey et al. [3] further categorized this dataset into three distinct aspects: function, data, and behavior. This classification scheme aids in systematically analyzing and understanding the requirements, enabling more informed research and development efforts.

2) Annotation Concerns Dataset: Based on the PROMISE NFR dataset, we conducted keyword extraction from the requirements text. Our approach was inspired by Glinz's method [5] to classify requirements. Glinz proposed a taxonomy that categorizes requirements based on concerns associated with each type of requirement. Our study simplified this taxonomy to differentiate between functional and non-functional requirements. Specifically, we considered performance requirements, specific quality requirements, and constraints as part of the non-functional requirements category.

For example, to annotate the requirement text, 'The system shall refresh the display every 60 seconds.'. The 'refresh the display' maps the functionality aspect, and the 'every 60 seconds' maps the time bounds—so we annotate these words as concerns.

The annotators employed the doccano annotation tool to annotate the requirement texts, and the resulting annotations were exported as JSON files. We have made the code and dataset available at the following link: [2]. You can visit the link provided to access the code and dataset associated with our study. To ensure the accuracy and consistency of the annotations, the annotation process involved two annotators who collaboratively reviewed and annotated 100 examples to reach a consensus on the labeling. Subsequently, they independently randomly annotated the remaining 525 examples. We employed the kappa statistic to calculate the inter-annotator agreement, which yielded a result of 0.7261, indicating a high level of agreement between the annotators. This joint effort further strengthened the reliability and consistency of the resulting annotations.

3) Train Extract Model: Considering the notable advances achieved by the Transformers-based BERT model, which has consistently demonstrated exceptional performance across various domains and tasks, we have chosen to fine-tune the BERT model specifically for the purpose of extracting concerns from requirements texts. By fine-tuning this powerful model, we aim to leverage its advanced capabilities and optimize its performance for our specific data extraction task.

To train our concerns extraction model, we utilized the code developed by Zhou et al. [19] from their previous work. We reutilized and fine-tuned their code for our specific task. We assessed our model's performance using a 10-fold cross-validation technique, which involves dividing the dataset into several subsets. This approach allowed us to train and test the model iteratively on different combinations of subsets, ensuring a comprehensive evaluation of its effectiveness and generalization capabilities.

To obtain a high-performance extraction model, we evaluated our model using precision (p), recall (r), and F1-score

[1]https://zenodo.org/record/268542

[2]https://drive.google.com/drive/folders/1jwolkzygPhO8VvbcE8JPWTi66G43KVJ9?usp=sharing

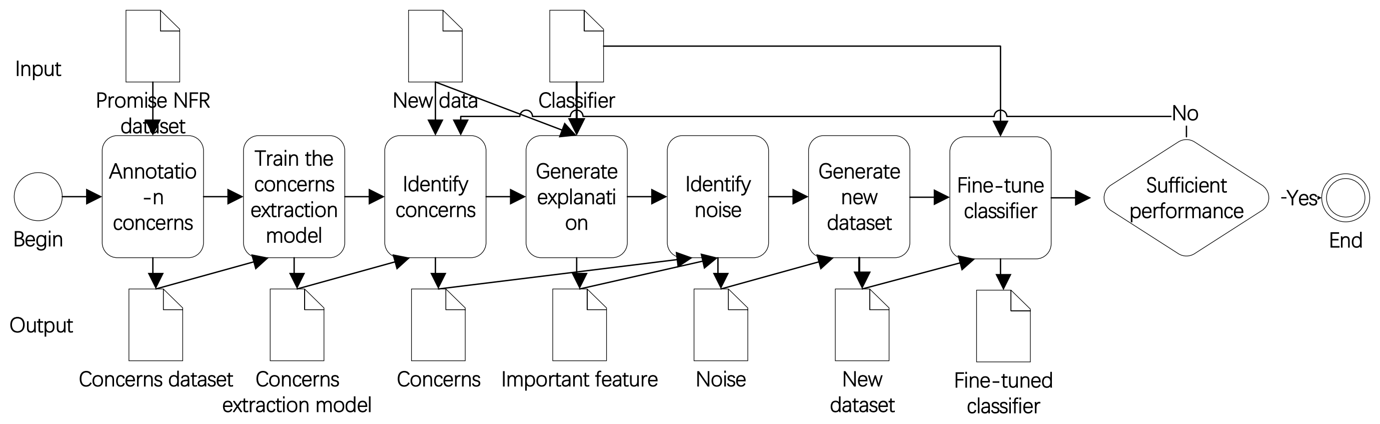

Fig. 1. Framework for Debugging Requirement Classification Model: Workflow Overview

metrics. The model demonstrated excellent performance, with an average precision of 0.88212, indicating its ability to accurately identify concerns from the requirements text. The recall value of 0.95512 signifies the model's capability to capture a high proportion of the real concerns in the text. Additionally, the F1 score of 0.91717 reflects a balanced combination of precision and recall, confirming the model's overall effectiveness in extracting concerns from the requirements text. These results validate the model's performance and suitability for our intended purpose.

B. Requirements Classification Model Evolution

We propose a noise detection technique to reveal the problems existing in the inference process of the requirements classification model. Specifically, we use SHAP as the explanation method to generate explanations. They are combined with the extraction results of the concerns extraction model to identify the noise. Finally, a new data set is generated by removing noise, and the original classification model is fine-tuned to improve model performance.

1) Generate Explanations: Our study employed the widely used explainability method, SHAP (Shapley Additive Explanations) [9], to generate explanations for our model's predictions. SHAP provides an essential value for each feature, indicating its contribution to a specific forecast. By analyzing these critical values, we gain insights into which features are genuinely significant and how they influence the model. This helps us understand the factors the model considers essential and the reasoning behind its predictions. SHAP has proven to be a valuable tool in interpreting and explaining the inner workings of machine learning models.

2) Identify Noise: In our explanation analysis, we observed that the importance of each feature could be positive or negative, indicating the direction and magnitude of its impact on the current classification. The absolute value of the importance score reflects the significance and influence of a particular feature on the model's decision-making process. A more significant absolute value indicates a more significant impact and suggests a higher level of importance for that

TABLE I
THE WORD IMPORTANCE OF REQUIREMENT TEXT 'THE SYSTEM SHALL REFRESH THE DISPLAY EVERY 60 SECONDS.'.

Word	IMPORTANCE FOR FR	IMPORTANCE FOR NFR
the	-0.0083	0.0083
system	0.0462	-0.0462
<u>shall</u>	0.5467	-0.5467
ref	0.2425	-0.2425
resh	-0.1532	0.1532
the	-0.0223	0.0223
display	0.0735	-0.0735
every	-0.3131	0.3131
<u>60</u>	-1.3088	1.3088
<u>seconds</u>	-1.4518	1.4518
.	0.0124	-0.0124

feature in the classification task. We consider the feature that absolute value over (1/N) is important. N is the number of words the requirements text contains.

Then we extract concerns from the requirements text using our trained extraction model. If an important feature, as determined by its high absolute value importance score, does not align with any of the extracted concerns, we consider it potentially noisy.

For example, the requirements document: 'The system shall refresh the display every 60 seconds.' is a non-functional requirement. We extract the concerns from the text. The result shows 'refresh,' 'display,' and 'every 60 seconds' are concerns. SHAP is used to generate explanations, and the importance of words is shown in Table I. The importance of words underlined exceeds the threshold and be considered important words. But the 'shall' is not a concern(based on the concern extraction model) and is regarded as a noise that will be removed in the new dataset.

3) Generate Enhanced Dataset And Model Evolution: Our approach implements a noise detection procedure on all examples in the PROMISE NFR dataset. We obtain a new dataset by identifying and removing noisy features from each sample. This new dataset is then used to fine-tune the original classifier to improve performance. To further improve the robustness of

TABLE II
THE OVERVIEW OF THE DATASET

	PROMISE NFR	**NFR_exp**
Functional requirements	255	189
Non-functional requirements	370	155
Total	625	344

our method, this process can be applied in multiple iterations. Each iteration involves detecting and removing noise from the previous dataset, resulting in a new dataset with improved data quality. This iterative approach allows us to generate multiple datasets, each with varying degrees of denoising. We refer to this process as model evolution.

IV. EVALUATION

A. Research Question

To evaluate our approach, we design three research questions.

RQ1: *Does the explanation-based model evolution lead to improved model performance?*

We would like to know whether this model evolution technique can improve the performance of the requirements classification model, to what extent, or why there is no performance improvement. Also, we would like to know the extent of improvement by conducting one iteration or a few time iterations.

RQ2: *To what extent is the utility of using concerns to detect noise?*

We wondered whether concerns help identify noise and whether stop words can replace them.

RQ3: *To what extent does the evolved model demonstrate generalization capabilities when applied to unseen datasets?*

We want to know whether there is a change in the generalization of the evolutionary model and whether there is an over-fitting phenomenon.

B. Dataset

In this study, we utilized two datasets to evaluate the performance of our proposed approach: PROMISE NFR and NFR_exp. (1) The first dataset, PROMISE NFR[3], consists of 625 samples, including 255 functional and 370 non-functional requirements. We named this dataset PROMISE NFR. (2) The second dataset, NFR_exp [20], comprises 344 samples, including 189 functional and 155 non-functional requirements. Our approach was tested on both datasets to evaluate its robustness and generalizability. Using these datasets allowed us to comprehensively evaluate our proposed approach's performance on diverse requirements sets. Table II presents an overview of the dataset.

C. Experiment Set And Execution

To answer RQ1, we trained a requirement binary classification model with the first dataset (PROMISE NFR) as the baseline. The model classified requirements into functional requirements and non-functional requirements.

[3]https://zenodo.org/record/268542

TABLE III
EVOLUTIONED MODEL EVALUATION RESULT ON PROMISE NFR AND NFR_EXP DATASET

	PROMISE NFR			**NFR_exp**		
Approach	P	R	F_1	P	R	F_1
Baseline	0.9076	0.9100	0.9083	0.6894	0.9097	0.7844
Evolution1	0.9290	0.9283	0.9287	0.6044	0.9444	0.7371
Evolution2	0.9648	0.9640	0.9644	0.6502	0.9166	0.7608
Evolution3	0.9844	0.9844	0.9844	0.6421	0.9097	0.7528

BERT (Bidirectional Encoder Representations from Transformers) is a powerful pre-trained language model known for capturing contextual information from text. It has demonstrated impressive performance in various natural language understanding tasks, showcasing its effectiveness in handling complex language processing challenges. With its advanced architecture and contextual representation learning, BERT has become a valuable tool in natural language processing. We use the Bert-base-uncased model to train the requirement classifier as the baseline.

We train it on the Google Colab platform with a T4 GPU. We used Precision (p), Recall (r), and F1-score metrics to evaluate our model's performance. Precision measures accuracy in identifying relevant concerns, Recall assesses the model's ability to capture genuine concerns, and F1-score evaluates overall effectiveness. These metrics provide a comprehensive understanding of our model's accuracy and completeness. We used a 10-fold cross-validation approach to evaluate our model's effectiveness. The findings reveal a mean precision of 0.9076, a recall rate of 0.9100, and an f1-score of 0.9088. Then we execute our approach on the dataset and classifier. We performed a total of three model evolutions. The evolution result is shown in Table III.

To answer RQ2, we compared the noise and stop-words list to determine their similarity. Specifically, we aimed to investigate whether the noise overlaps significantly with the stop-words. By calculating the ratio of stop-words within the noise, we obtained a result of 0.26032. This finding indicates that stop-words constitute only a small subset of the noise present in the dataset. Therefore, it can be inferred that the noise encompasses a broader range of elements beyond just the stop-words. This analysis provides valuable insights into the distinct nature of noise and stop-words, highlighting the need for dedicated noise detection techniques beyond solely relying on stop-word removal.

For answering RQ3, we examined the second dataset (NFR_exp) to assess the potential impact of overfitting and generalization in our technique. This dataset, provided by Lima et al. [20], expands upon the first dataset (PROMISE NFR) with an additional 347 requirements text samples. We utilized these new samples to evaluate the generalization ability of the baseline classifier and the three evolved models. Table III presents the performance results of these models on the newly augmented dataset.

D. Result

We evaluated the model evolution technique based on the research questions, and we will answer these questions in the following with the evaluation results.

RQ1:Does the explanation-based model evolution lead to improved model performance? To answer RQ1, we trained a requirement classifier as the baseline and used the approach to evaluate performance improvement. The result shows the approach can improve the F_1 by 8 percent. Our approach demonstrates its effectiveness in denoising the dataset and enables the model to focus on learning the essential features, thereby improving the overall performance of the classification model.

RQ2:To what extent is the validity of using concerns to detect noise? To answer RQ2. By comparing noise and points of interest to assess its accuracy and usefulness. The results revealed that noise encompasses not only stop words but also other irrelevant or non-deterministic words. This finding validates the effectiveness of our approach in correctly identifying and distinguishing noise from meaningful information in the dataset.

RQ3:To what extent does the evolved model demonstrate generalization capabilities when applied to unseen or external datasets? To answer RQ3. We conducted our approaches and comprehensively evaluated our noise identification technique on another dataset. We evaluate the original model and three evolved models on the PROMISE NFR expansion dataset. The results show that our technique slightly reduces the model's generalization performance, possibly due to overfitting due to the high similarity between the new data generated by denoising and the original data. In any case, our approach provides an idea further to improve the model's performance on the same dataset.

E. Threats To Validity

In this section, we discuss the possible threats to the validity of our research and experimental design.

Construct Validity. Due to the lack of labeled requirements documents, we only use the PROMISE NFR dataset to train the extraction models of concerns. This approach may lead to false noise identification. In addition, we use widely used evaluation metrics and fixed parameter settings to ensure the reproducibility of experiments.

Internal Validity. To alleviate the risk of overfitting, we implemented the early stopping method during the training process of the baseline classifier. This technique allowed us to monitor the model's performance on a separate validation set and halt the training when the performance ceased to improve. Moreover, we ensured the generalizability of our results by randomly splitting the dataset, allocating 90% for training and the remaining 10% for testing. We have taken the initiative to publicly share all the codes used in our experiments. By making these resources accessible, we aim to facilitate the research community's replication and verification of our results.

External Validity. Considering the differences caused by the environment, we ran the experiment on Colab and disclosed the code to ensure the experiment's reproducibility. In addition, we choose the widely-used BERT model as the baseline.

Conclusion Validity. To avoid possible errors during the experiments, we used widely used model training procedures and datasets and performed them jointly with multiple participants.

V. Discussion

In this section, we discuss several potential issues and the meaning of our approach.

The results indicate that our approach may have a slight impact on the generalization of the model. One possible reason is overfitting, as the denoised data generated closely resembles the original data. Another factor could be the strong performance of our baseline model itself, which may result in less pronounced effects of the evolutionary model. Additionally, the incorrect concerns extracted by the concerns extraction model pose a challenge in accurately identifying noise.

Nevertheless, our approach offers valuable insight for improving the model performance in the same dataset. Furthermore, it sheds light on the reasoning process of black-box models through explainability analysis. By exposing model flaws, we can optimize the model more effectively. In the future, we want to test our approach on additional datasets and improve its ability to identify noise accurately.

VI. Conclusion

In this paper, we have addressed the critical problem of debugging requirement classification models in software engineering. We have presented an approach that combines concern extraction and model evolution based on explainability to improve the transparency and performance of requirement classification models.

First, we trained a concern extraction model using the PROMISE NFR dataset and BERT. The model can extract concerns from requirement texts, providing valuable insights into the decisive factors in requirement classification. Next, we employed SHAP, a widely-used explainability method, to generate explanations for the predictions of the requirement classification model. We identified important features and potential noise in the model's decision-making process by analyzing these explanations. We identified potential noise in the model's predictions by analyzing important features and comparing them with extracted concerns. This noise can indicate areas where the model's decision-making process may be unreliable or inconsistent with the intended classification. To improve the model's performance and reliability, we created a new dataset by removing the identified noise to improve the model's performance and reliability. This dataset was then used to fine-tune the original requirement classification model, resulting in an improved model that is more robust and accurate in its predictions.

In conclusion, combining concern extraction and model evolution based on explainability offers a promising approach to debugging requirement classification. By understanding the underlying reasoning and addressing potential noise, we can improve the accuracy and trustworthiness of requirement classification models, ultimately resulting in better software systems.

Acknowledgment

This work is partially supported by the Project of Beijing Municipal Education Commission (No.KM202110005025) and the National Natural Science Foundation of China (No.62162051).

References

[1] M. Binkhonain and L. Zhao, "A review of machine learning algorithms for identification and classification of non-functional requirements," *Expert Systems with Applications: X*, vol. 1, p. 100001, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590188519300010

[2] M. Lu and P. Liang, "Automatic classification of non-functional requirements from augmented app user reviews," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE'17. New York, NY, USA: Association for Computing Machinery, 2017, p. 344–353. [Online]. Available: https://doi.org/10.1145/3084226.3084241

[3] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy, "Norbert: Transfer learning for requirements classification," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 2020, pp. 169–179.

[4] J. Winkler and A. Vogelsang, "Automatic classification of requirements based on convolutional neural networks," in *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, 2016, pp. 39–45.

[5] M. Glinz, "On non-functional requirements," in *15th IEEE international requirements engineering conference (RE 2007)*. IEEE, 2007, pp. 21–26.

[6] C. B. Azodi, J. Tang, and S.-H. Shiu, "Opening the black box: interpretable machine learning for geneticists," *Trends in genetics*, vol. 36, no. 6, pp. 442–455, 2020.

[7] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[9] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[10] S. Ezzini, S. Abualhaija, C. Arora, and M. Sabetzadeh, "Taphsir: Towards anaphoric ambiguity detection and resolution in requirements," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 1677–1681. [Online]. Available: https://doi.org/10.1145/3540250.3558928

[11] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Requirements classification with interpretable machine learning and dependency parsing," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, pp. 142–152.

[12] A. Bibal, M. Lognoul, A. De Streel, and B. Frénay, "Legal requirements on explainability in machine learning," *Artificial Intelligence and Law*, vol. 29, no. 2, pp. 149–169, 2021.

[13] P. Lertvittayakumjorn and F. Toni, "Explanation-based human debugging of nlp models: A survey," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1508–1528, 2021.

[14] M. Hartmann and D. Sonntag, "A survey on improving nlp models with human explanations," *arXiv preprint arXiv:2204.08892*, 2022.

[15] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International conference on machine learning*. PMLR, 2017, pp. 1885–1894.

[16] H. Zylberajch, P. Lertvittayakumjorn, and F. Toni, "Hildif: Interactive debugging of nli models using influence functions," in *Proceedings of the First Workshop on Interactive Learning for Natural Language Processing*, 2021, pp. 1–6.

[17] P. Lertvittayakumjorn, L. Specia, and F. Toni, "Find: human-in-the-loop debugging deep text classifiers," *arXiv preprint arXiv:2010.04987*, 2020.

[18] J. Cleland-Huang, S. Mazrouee, H. Liguo, and D. Port, "nfr," Mar. 2007. [Online]. Available: https://doi.org/10.5281/zenodo.268542

[19] Q. Zhou, T. Li, and Y. Wang, "Assisting in requirements goal modeling: A hybrid approach based on machine learning and logical reasoning," in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 199–209. [Online]. Available: https://doi.org/10.1145/3550355.3552415

[20] M. Lima, V. Valle, E. a. Costa, F. Lira, and B. Gadelha, "Software engineering repositories: Expanding the promise database," in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, ser. SBES '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 427–436. [Online]. Available: https://doi.org/10.1145/3350768.3350776