Enhancing ML Prediction Model In Requirement Engineering

Sulbha Malviya
Department of Computer Science
Boise State University
sulbhamalviya@u.boisestate.edu

Mehenaz Afrin
Department of Computer Science
Boise State University
mehenazafrin@u.boisestate.edu

I. Introduction

In recent years, machine learning (ML) has become a critical component in software systems, but concerns about the bias of these models continue to rise, especially in requirements engineering. Previous studies have introduced frameworks to address fairness in the later stages of ML development, but methods for early-stage fairness analytics, such as identifying sensitive features from user stories, remain underdeveloped. This research builds upon and extends prior work by addressing the limitations of existing fairness frameworks, particularly ReFair [1], which uses shallow machine learning models to classify sensitive features. While ReFair demonstrated promising results with simple models, it left a gap in the exploration of deep learning techniques that could potentially offer higher accuracy and more nuanced insights.

Our proposed research aims to bridge this gap by introducing advanced word embedding techniques and deep learning models for classifying sensitive features from user stories. This approach seeks to enhance the accuracy and contextual understanding in the early stages of machine learning development, providing a more sophisticated solution for identifying biases in requirement engineering processes. By employing deep learning models, we aim to surpass the performance of simpler models, addressing the unexplored potential in this area and contributing to a more robust and fair ML pipeline. [2].

Problem Description

Machine learning models, particularly those used in requirement engineering, may overlook important contextual clues or focus on irrelevant patterns when processing complex textual data, such as user stories. This can lead to less accurate predictions or biased decisions, especially when the training data is imbalanced or fails to adequately represent all variations of user stories. Addressing these challenges is critical to ensuring fairness and accuracy in early-stage machine learning applications within software development.

Problem Importance

The problem of bias in machine learning (ML) models during the requirements engineering phase is both underexplored and critical. Addressing biases early in the software development lifecycle is essential to avoid perpetuating unfairness in system behavior. This section outlines the key challenges that motivate the need for more advanced solutions in this space:

- **Limited Research Work:** Despite the growing integration of machine learning in software development, very few studies have focused on the identification of biases during the early phases, such as requirements specification and analysis. Most existing research on ML fairness focuses on model training or post-deployment monitoring, leaving a significant gap in the early stages of development. This lack of focus on initial stages is particularly concerning since bias introduced early can propagate through the entire system.
- **Current Frameworks Use Shallow ML Models:** The current state-of-the-art frameworks for detecting sensitive features within user stories typically employ shallow machine learning models [1]. These models, which consist of only a few layers, are simpler and computationally less expensive but may lack the capacity to capture complex patterns in the data. Shallow models often struggle to perform well when it comes to nuanced classification tasks that require a deeper contextual understanding, which is critical in requirements engineering.
- **Avoidance of Advanced AI Solutions:** One reason for the reliance on shallow models in previous frameworks is the concern over the computational cost and reduced interpretability associated with advanced AI techniques, such as deep learning. While these concerns are valid, the avoidance of deep learning techniques limits the effectiveness of these frameworks in terms of capturing context-specific insights. Advanced models, such as those utilizing deep learning and word embeddings, have the potential to uncover deeper biases and improve fairness in machine learning systems, yet they remain underexplored in this domain.

Proposed Solution

Current Framework

The current framework [1] for identifying sensitive features from user stories has laid a strong foundation, but

it still exhibits several limitations that restrict its full potential in handling more complex real-world scenarios. The key components of the current framework are:

- **Shallow Machine Learning Algorithms:** The current approach employs shallow machine learning models, such as Decision Trees, Random Forests, and Support Vector Machines (SVMs). These models are computationally efficient and interpretable but struggle with capturing complex patterns and contextual nuances within user stories. In particular, shallow models may overlook deeper semantic relationships between words or phrases, reducing their accuracy in classifying tasks and application domains.
- **Limited Exploration of Advanced Models:** While the framework has achieved notable results with shallow models, it has not fully explored more advanced machine learning techniques, such as deep learning architectures. Deep learning models are known for their ability to capture hierarchical and contextual representations, which are crucial in analyzing textual data like user stories. Without the use of these advanced models, the framework's ability to detect and mitigate bias remains limited.

Proposed Framework

To address these limitations and improve the fairness and accuracy of the framework, we propose several enhancements that involve the adoption of more advanced classification techniques. These improvements are designed to make the framework more robust, generalizable, and effective in reducing biases in the early stages of software development.

Advanced Word Embedding Techniques:

- **Variants of Bert:** We propose leveraging advanced word embedding techniques, specifically by utilizing variants of BERT, such as Albert,RoBERTa and DistilBERT. These models generate contextually rich embeddings that capture subtle linguistic and semantic nuances within user stories.

  The use of such embeddings enhances the model's ability to understand complex language structures, improving classification accuracy. These embeddings are particularly useful for identifying sensitive features and reducing the risk of biased decisions stemming from inadequate language understanding.

Advanced Deep Learning Approaches:

- **Deep Learning Architectures:** To address the limitations of traditional and shallow models, we propose leveraging advanced deep learning architectures, including Long Short-Term Memory (LSTM) networks, Bidirectional LSTM (BiLSTM), Gated Recurrent Unit (GRU) networks, and Bidirectional GRU (BiGRU). These architectures are particularly well-suited for handling sequential data such as user stories, offering the ability to capture both short-term and long-term dependencies effectively.
  - **LSTM and BiLSTM:** LSTMs are capable of retaining information over extended sequences, making them ideal for understanding the contextual and temporal relationships inherent in user stories. BiLSTMs extend this capability by processing the data in both forward and backward directions, enabling the model to grasp contextual relationships more comprehensively.
  - **GRU and BiGRU:** GRUs simplify the architecture of LSTMs while retaining their ability to manage long-range dependencies. BiGRUs further enhance this by considering the sequential data from both past and future contexts, providing improved generalization and classification accuracy.
- **Improved Feature Representation:** By employing these deep learning models, the framework can process and represent sensitive features with higher precision. These architectures can automatically learn meaningful patterns, dependencies, and semantic nuances in the input text, leading to better classification outcomes.
- **Robust Bias Detection and Generalization:** The adoption of these architectures makes the framework more robust against noise in the data and enhances its ability to generalize across diverse scenarios. These improvements are instrumental in reducing biases in early-stage software development, ensuring more fair and accurate predictions.

The proposed framework aims to enhance the identification of sensitive features from user stories in the context of machine learning. It begins with user stories as input, which undergo preprocessing to ensure they are suitable for analysis.

Next, advanced word embedding techniques, such as ALBERT, RoBERTa, and DistilBERT, are employed to generate contextually rich embeddings. These embeddings improve the model's understanding of nuanced language, enabling a more accurate classification of the machine learning task and the associated application domain.

Following the classification phase, the framework extracts sensitive features specific to both the domain and the machine learning task. The intersection of these sensitive features results in a set of recommended sensitive features that are essential for story development. This structured approach aims to minimize biases and improve the fairness of machine learning models in requirements engineering.

Overall, our proposed framework aims to significantly enhance the fairness and accuracy of sensitive feature detection in user stories by leveraging both advanced word embeddings and deep learning techniques. These improvements will enable practitioners to address biases earlier in the software development lifecycle, resulting in more equitable and trustworthy machine learning systems.

II. Related work

Rahman et al. [3] designed a tool named "GENEus" by using GPT4.0 for generating user stories automatically from

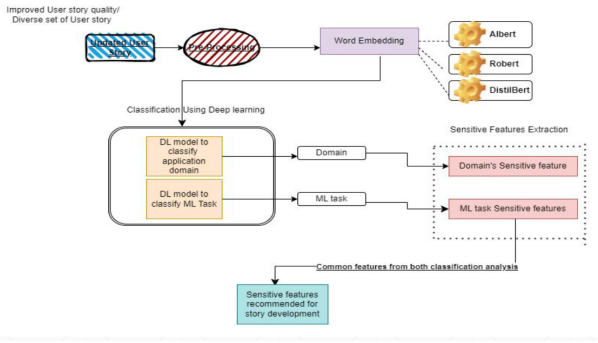

Fig. 1. Proposed Framework for Analysing User Stories

requirements documents to significantly reduce the time and effort of developers, leading to productivity. The output result from this tool is in JSON format, helping smooth integration with the latest project management platforms. To accurately create user stories and avoid hallucination problems in the (LLM) Large Language Model, they adopt the Refine and Thought (RAT) strategy which discarded meaningless tokens and remove the redundancy form LLM, thus create reliable user story and test cases. They make survey questionnaires using RUST (Readability, Understandability, Specifiability, Technical aspects) which sent over to 50 developers from different backgrounds to validate their research outcomes. Despite their appreciated quality score, there is room for potential improvement by adding domain knowledge to reduce the hallucination problems. Also, researchers can work on prompt refinements and continuous testing for the enhancement. They can also collect documents from various large software products to generalize their work. Manual developers ensure that RAT sometimes deletes the crucial part of the information.

To alleviate plagiarism, better understanding, and enhance reading quality AZIMOV et al. [4] emphasize on paraphrasing task of user stories. By providing instructions, they try to evaluate the accuracy of 10 types of LLMs regarding the manipulation of user stories. In this process, they use 23 stylometry metrics from existing research work for identifying bots in the writing style. To conduct their research work, they fetch the dataset containing user stories from Ferrara et al.'s [1] Refair study. Each LLM was involved in paraphrasing 100 user stories and the author did a comparative analysis between the paraphrased version and the original version. This analysis shows that the difference between these versions is trivial. Among all the LLM models, LLaM3 performs better than others in terms of stylometric management. Their work can be extended by testing larger models of LLaM3 along with 70 billion parameters to get an idea of how it will work on large projects. Additionally, to ensure generalizability and versatility, future researchers can measure the accuracy of various datasets. They can also use a variety of matrices to examine detailed model behaviors.

Herwanto et al. [5] offers a convincing method for incorporating Natural Language Processing (NLP) into agile software development's privacy requirements engineering (PRE) process. The automation of the process of extracting privacy requirements from user stories solves important problems related to manual, traditional PRE approaches that are unable to keep up with agile methodology. One noteworthy contribution is the creation of a graphical user interface (GUI) tool to let requirement engineers and automated systems collaborate. While the paper successfully illustrates how NLP can increase efficiency, it would benefit from a deeper examination of the drawbacks and potential biases present in NLP models as well as a more concise articulation of the empirical data demonstrating the viability of the suggested strategy. To enhance the case overall, it would also be beneficial to expand on the subject of how the tool can adjust to changing privacy legislation and the real-world applications it may be used for. In the subject of privacy engineering, ML-based solutions can automate the process of choosing the best design solutions for every privacy requirement.

The difficulties of requirements management are examined by Jukka Kääriäinen et al. [6] in the context of Extreme Programming (XP), an agile approach intended to improve software development's responsiveness and flexibility. They contend that although XP encourages minimal documentation and customer participation, it also has challenges when it comes to efficiently handling tasks and user stories, especially in dynamic contexts where conventional requirements management techniques could utilize too many resources. In order to preserve the agile XP concepts while enabling electronic user story management and storage, the suggested solution, StoryManager, attempts to combine requirements management and configuration management within the Eclipse environment. The authors stress how crucial it is to modify tools to meet project-specific requirements while maintaining XP's agile principles. On the other hand, a more thorough examination of empirical findings and user comments about StoryManager's performance in practical applications will enhance the study. Furthermore, even if the integration of requirements and implementation represents a major advancement, the suggested solution's wider application may be constrained by its dependence on the Eclipse development environment. It raises important questions about the balance between maintaining agility and ensuring adequate requirements traceability, suggesting that further research is needed to validate the proposed solutions and explore their adaptability across different development environments.

For ensuring the quality of user stories, Lucassen et al [7] created Automatic Quality User Story Artisan (AQUSA) software tool by applying natural language processing. This framework contains 13 criteria which every writer should follow to improve the quality of the user story. if any user story need improvement, their tool will detect it as poor quality and suggest solutions.

Raharjana et al. [8] invented a conceptual model for enhancing domain knowledge comprehension by extracting user

stories from online news. The news contains lessons learned information associated with certain events. While other researchers are trying to get user stories from NLP applications, graphical conceptual models, and UMLs, this author focuses on unstructured documents from several sources which enhance the efficiency of requirements elicitation and ensure domain knowledge improvement to assist system analysts. They preprocess their dataset by using stop word removal, tokenizer, WordNet lemmatization, and parts of speech tagging. Their result reveals that this model extracts 105 user stories and precisely detects 8 aspects of why, 41 aspects of who, and 94 aspects of what based on their case study. Future researchers can employ this model by using different and massive online news to check if this method can extract user stories accurately or not.

Limited research has been done in the requirement engineering for the assessment of machine learning model fairness. A recent study emphasizes this topic but still, there are unexplored areas such as deep learning exploration, and large-scale user stories [1]. To bridge the previous gap, we aim to extract sensitive features to remove the biases in the prediction level by leveraging deep learning algorithm along with advanced word embedding techniques on the large-scale dataset.

III. Approach

A. Transformer-based Models: ALBERT, RoBERTa, and DistilBERT

Transformer-based models like ALBERT, RoBERTa, and DistilBERT are leveraged for advanced embedding generation. These models represent the state-of-the-art in natural language understanding and are known for their ability to handle long-range dependencies and contextual word meanings. Here's how they are utilized:

- **ALBERT (A Lite BERT)**: This model reduces memory consumption and improves efficiency while maintaining performance [9]. ALBERT is employed to obtain the word embeddings and contextual features from the user stories.
- **RoBERTa (Robustly Optimized BERT Pretraining Approach)**: RoBERTa enhances the original BERT by training it on more data and with more hyperparameter tuning [10]. It is used for generating embeddings that are particularly effective in capturing the semantics of user stories.
- **DistilBERT**: A distilled version of BERT, which offers a trade-off between speed and accuracy, DistilBERT is used to balance computational efficiency with performance [11].

These transformer models are fine-tuned on the specific domain of software development and integrated into the classification pipeline.

B. BiLSTM, GRU, and BiGRU Models for Sequential Data

Once the word embeddings are obtained, we use advanced recurrent neural network (RNN)-based architectures to process the sequential nature of the input data. The models used include:

- **BiLSTM (Bidirectional Long Short-Term Memory)**: LSTM networks are well-suited for sequential data because they capture long-range dependencies. The bidirectional variant (BiLSTM) improves this further by processing the sequence in both forward and backward directions, allowing the model to fully understand the context at each time step [12].
- **GRU (Gated Recurrent Unit)**: GRUs are similar to LSTMs but are computationally simpler and faster. They are capable of capturing long-term dependencies and have fewer parameters, making them a more efficient choice in certain scenarios [13].
- **BiGRU (Bidirectional GRU)**: Similar to BiLSTM, the BiGRU processes data in both directions, leveraging the benefits of the GRU architecture with bidirectional context [14].

These architectures are applied to the word embeddings generated by ALBERT, RoBERTa, and DistilBERT to learn complex temporal patterns and relationships in the text, improving the model's ability to classify user stories.

These results highlight the effectiveness of hybrid approaches for sensitive feature extraction, with RoBERTa and Label Powerset proving particularly impactful.

IV. Implementation

We conducted several experiments to combine transformer-based models such as ALBERT, RoBERTa, and DistilBERT with various RNN-based architectures, including BiLSTM, GRU, and BiGRU, to classify Domain and ML Task. The details of the model architecture and training process are described below.

A. Model Architecture for Domain Classification

In this part, we have focused on detecting domains from user stories by leveraging neural network architectures designed for sequence modeling. The following models were explored:

Gated Recurrent Unit (GRU) Model:

- **Embedding Layer:** Converts input tokens into dense vectors of dimension 128, initialized using the Glorot Uniform initializer.
- **First GRU Layer:** Contains 64 units and returns sequences to pass the output to the next recurrent layer.
- **Second GRU Layer:** Contains 32 units and processes the sequence output from the first GRU layer.
- **Dense Output Layer:** A fully connected layer with a softmax activation function to predict domain classes.
- **Learning Rate:** The Adam optimizer is used with a learning rate of 0.001. The model is compiled with a categorical cross-entropy loss function and accuracy as the performance metric.
- **Initialization:** The Glorot Uniform initializer is applied to ensure balanced weight distribution at the start of training.

Bidirectional GRU (BiGRU) Model for Domain Detection: BiGRU architecture is designed to capture contextual information from both past and future sequences by processing the input in both forward and backward directions. This makes it particularly effective for tasks like domain detection, where context plays a crucial role.

- **Embedding Layer**
  - Purpose: Converts input tokens into dense fixed-size vectors (128 dimensions).
  - Initializer: GlorotUniform is used to ensure balanced weight initialization.
  - Input Dimension: vocab-size + 1 (accounting for padding token).
- **Bidirectional GRU Layers**
  - **First BiGRU Layer:**
    - 64 GRU units in each direction (forward and backward), resulting in 128 combined outputs.
    - return-sequences=True ensures that the entire sequence is passed to the next layer.
  - **Second BiGRU Layer:**
    - 32 GRU units in each direction, providing 64 combined outputs.
    - Processes the output from the first BiGRU layer.
  - **Dense Output Layer**
    - A fully connected layer with softmax activation to produce class probabilities.
    - The number of neurons corresponds to num-classes, representing the possible domains.
  - **Maximum Sequence Length:** Input sequences were padded to a maximum length of 100 to standardize the input dimensions across samples.

Bidirectional Long Short-Term Memory (BiLSTM) Model: The BiLSTM architecture followed a similar structure, replacing BiGRU layers with bidirectional LSTM layers:

- Embedding Layer: Same as the GRU model.
- BiLSTM Layers: Two stacked bidirectional LSTM layers, each processing the input in both forward and backward directions.
- Dense Output Layer: Similar softmax layer for classification.

B. Model Architecture for ML Task Classification

Each RNN-based model is constructed with the following layers:

- **Embedding Layer:** The input text is tokenized, and the transformer model (e.g., DistilBERT, Albert, Robert) generates word embeddings. These embeddings are passed through an embedding layer in the BiLSTM, GRU, or BiGRU models for further processing.
- **RNN Layer:** The core of each model is a bidirectional RNN (BiLSTM, GRU, BiGRU) layer, which is responsible for capturing long-range dependencies in the input sequences. Bidirectional RNNs process the input sequence in both forward and backward directions, allowing the model to learn context from both past and future words. This is particularly useful for understanding the full context of the text.
- **Dropout Layer:** A dropout layer is applied after the RNN layer to prevent overfitting. Dropout randomly drops some units during training, forcing the model to learn more robust features by reducing its reliance on specific units.
- **Fully Connected (FC) Layer:** The output from the RNN layer is passed through a fully connected (linear) layer, which maps the high-dimensional RNN output to the desired output size. This layer makes the final predictions for the classification task.
- **Sigmoid Activation:** Since the task is multi-label classification, a sigmoid activation function is used in the output layer. This function outputs probabilities between 0 and 1, indicating the likelihood of each label being true for a given input text.

The final output layer produces probabilities for each class, where each probability corresponds to the likelihood of a label being assigned to a given input text.

$$\hat{y} = \sigma(Wh + b)$$

[15]

Where:

- $\hat{y}$ is the predicted output for each label.
- σ represents the sigmoid activation function.
- W and b are learned parameters (weights and biases).
- h is the output from the RNN layer (either BiLSTM, GRU, or BiGRU).

C. Training and Evaluation

1) Loss Function: The model is trained using the binary cross-entropy loss function, which is suitable for multi-label classification tasks. The binary cross-entropy loss is given by:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

[16]

Where:

- N is the number of samples in the batch.
- y_i is the true label for the i-th sample.
- $\hat{y}_i$ is the predicted probability for the i-th sample.

This loss function allows the model to calculate the error between the predicted and actual labels and update the weights accordingly.

2) Optimizer: The optimizer used for training the models is Adam, a popular optimization algorithm that adapts the learning rate during training. Adam is combined with a learning rate of 0.001 and weight decay for regularization. The weight decay is applied to prevent overfitting by penalizing large weights. The Adam update rule is as follows:

$$\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{v_t} + \epsilon} + \lambda \theta_t$$

[17]

Where:

- θ_t represents the model parameters at time step t.
- η is the learning rate.
- m_t and v_t are estimates of the first and second moments of the gradients.
- λ is the weight decay factor.
- ϵ is a small constant to prevent division by zero.

3) Training Process: The training process involves feeding tokenized text sequences into the transformer model to generate embeddings, which are then passed to the RNN layers for sequence modeling. The output of the RNN layers is passed through the dropout layer, fully connected layer, and finally a sigmoid activation to predict the probabilities for each label. The binary cross-entropy loss function is used to compare the predicted labels to the true labels, and the model parameters are updated using the Adam optimizer.

Each model was trained for a multiple number of epochs (e.g., 8 ,20,50 epochs) for ML classification. The training was monitored by evaluating the model on a validation set after each epoch, ensuring that the model generalizes well to unseen data.

D. Experiments

The following transformer + RNN combinations were tested:

- ALBERT + BiLSTM, GRU, BiGRU
- RoBERTa + BiLSTM, GRU, BiGRU
- DistilBERT + BiLSTM, GRU, BiGRU

Each experiment was evaluated based on its performance in the multi-label classification task. The evaluation metrics used to assess the models include F1-score, and hamming loss, which provide a comprehensive measure of model performance, particularly in the case of multi-label datasets. For model classification, precision, recall, f1 score, and accuracy, these evaluation metrics are used to assess the deep learning models. F1 score and accuracy are only used to assess 25 machine learning classifiers.

By combining transformer-based models with RNN architectures, this approach leverages the strengths of both types of models for processing sequential text data. The experiments show that the choice of RNN architecture, along with the transformer model, influences the model's performance on multi-label classification tasks. These results highlight the potential of combining transformer and RNN architectures to handle complex sequence classification tasks in software development.

V. Evaluation

In this section, we present the evaluation results Initially based on test size and later based on the k-fold cross-validation technique. The models were evaluated using metrics such as Precision, Recall, F1 Score, and Hamming Loss, Accuracy. The following paragraphs summarize the performance of the different models.

Research Questions

Our proposed model is evaluated based on 4 research questions. These 4 research questions are situated following:

RQ1:Do Bert versions enhance the performance level in classifying ML-specific application domains from user stories?

RQ2:Do Bert versions enhance the performance level in classifying ML-specific tasks from user stories?

RQ3:Can the application of Deep learning approaches higher the performance level in classifying ML-specific application domains from user stories?

RQ4: Can the application of Deep learning approaches higher the performance level in classifying ML-specific tasks from user stories?

Results From Domain Classification

In this section, we will describe how RQ1 and RQ3 are satisfied. For the domain classification Initially, we set the test size to 0.5 and then measured the performance level of 25 ML classifiers with different word embedding techniques. With 0.5 size, we tried 125 possible combinations to see which combination gave us the best result. We have used lazy predict which is a Python library designed to simplify the process of trying out multiple machine-learning models quickly [18]. When working on classification tasks, the Lazy Predict Classifier in this library helps us evaluate a wide range of classification algorithms without writing extensive code. This allows us to compare different models' performance and select the best one for your dataset.

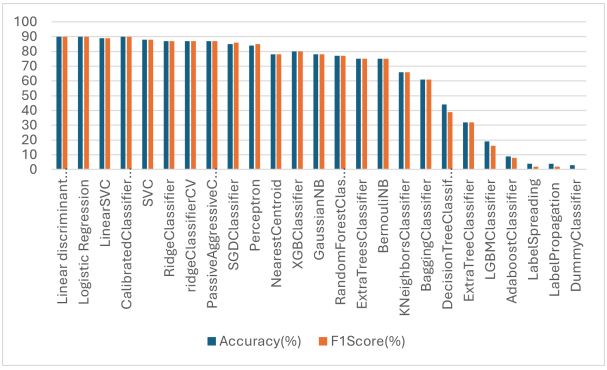

Fig. 2. Performance Level of Glove with 25 ML classifier in terms of Accuracy and F1 Score

These 25 ML classifiers are namely XGB Classifier, Bagging Classifier, Decision Tree Classifier, Extra Trees Classifier, Random Forest Classifier, Logistic Regression, LinearSVC, CalibratedClassifierCV, ExtraTreeClassifier, LinearDiscriminantAnalysis, SVC, SGDClassifier, BernoulliNB, Perceptron,PassiveAggresiveClassifier, NearestCentroid, RidgeClassifierCV, RidgeClassifier, RidgeClassifier, QuadraticDiscriminantAnalysis, KNeighborsClassifier, LabelPropagation, Label-

Spreading, LGBMClassifier, GaussianNB, AdaBoost Classifier, Dummy Classifier. We try to combine those classifiers with Glove, Bert, Albert, Distil Bert,and Roberta.

From Figure 2 we can see among all the classifiers with Glove, Linear regression, Logistic Regression, and Linear SVC give us the best and similar results by achieving 90% accuracy and F1 score. In this case, the Bagging classifier shows 60% performance level and the XGB classifier gives an 80% performance level both in accuracy and f1 score. Strangely logistic regression here performs better than the boosting and bagging classifiers.

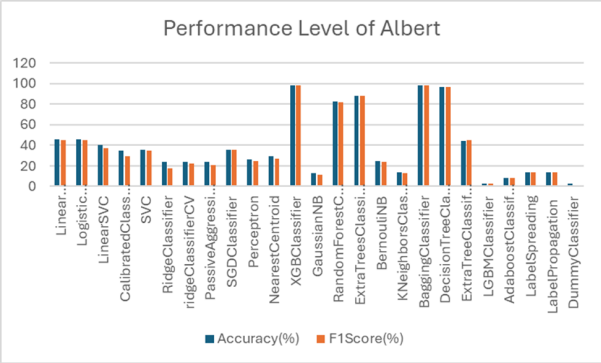

Fig. 3. Performance Level of Albert with 25 ML Classifier in terms of Accuracy and F1 Score

Figure 3 demonstrates how Bert's version works with different classifiers. Linear regression, logistic regression, and linear SVC decrease their performance level with Bert versions while they outperform by using Glove. But with Bert versions, the XGB classifier, Bagging classifier, and decision tree classifier show the highest result by giving 98% accuracy and f1 score in the detection of the domains from user stories. The dummy classifier shows no accuracy. After that, we employed a 10-fold cross-validation technique while determining performance level with AlBert, DistilBert, and Roberta along with those ML classifiers. We have shown the three best classifiers among 25 classifiers for classifying domains.

Distil Bert		
Classifier	Accuracy	f1 score
XGBClassifier	99	99
BaggingClassifier	98.6	98.2
DecisionTreeClassifier	98.3	98.3

Fig. 4. Performance level of ML classifiers with Distil Bert after applying 10-fold cross-validation technique.

After applying a 10-fold cross-validation technique, among all the Bert versions and classifiers, we get the best result with Distil Bert (figure 4) along with XGB, Bagging, and Decision tree Classifiers. XGB classifiers outperform Distil Bert showing 99% accuracy and f1 score while Bagging classifiers give 98.6% and 98.2% respectively in accuracy and f1 score. Decision Tree classifiers achieve a 98.3% performance level both in accuracy and f1 score.

Column1	Column2	Column3	Column4	Column5	Column6	Column7
Their framework				Our framework		
Classifier	Accuracy	F1 Score		Classifier	Accuracy	F1 Score
XGBClassifier	98	98		XGBClassifier	99	99
BaggingClassifier	98	98		BaggingClassifier	98.6	98.2
DecisionTreeClassifier	98	98		DecisionTreeClassifier	98.3	98.3

Fig. 5. comparison between their framework and our framework with ML classifiers

The figure 5 helps us satisfy our RQ1. Ferrara et al. get good results 98% accuracy and f1 score in the XGB classifier, Bagging Classifier, and Decision Tree Classifier by using the Bert technique. To enhance the performance level, we have used several advanced Bert variants. Among all the Bert variants, Distil Bert shows the highest performance level with the XGB, Bagging, and Decision Tree classifier. XGB classifier along with Distil Bert 99% accurately able to classify domains from user stories. Distil Bert helps to enhance performance levels in classifying domains. Thus, our RQ1 satisfied here by enhancing the performance level in compared to the previous framework. The previous framework only used shallow machine learning classifiers. They did not explore various Deep-learning techniques. But in this study, we have also explored a variety of RNN models to see if this helps to higher the classifying accuracy or not. For domain classification along with advanced word embedding techniques, we have applied BiLSTM, BiGRU, and GRU models.

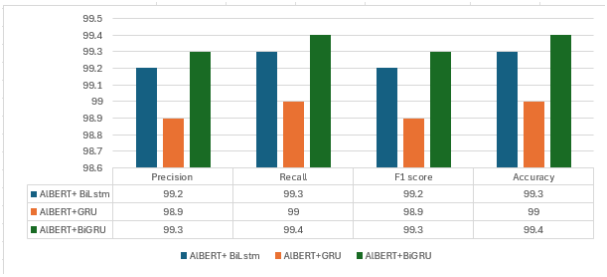

Fig. 6. Performance levels of Deep learning models with Albert in terms of Accuracy, Precision, Recall, and F1 score.

Figure 6 shows us ALBERT + BiGRU achieves the highest performance across all metrics, with Precision, Recall, F1 Score, and Accuracy being consistently high (around 99.3 to 99.4). ALBERT reduces the number of parameters while maintaining performance, which enhances efficiency during training and inference [19]. ALBERT + BiLSTM also performs well, achieving 99.2 in Precision and F1 Score, with 99.3 in Recall and Accuracy. ALBERT + GRU shows the lowest performance in this group, with scores between 98.9 and 99.0. GRUs lack the bidirectional context of BiLSTM and BiGRU, which reduces their ability to capture nuanced relationships in both directions. GRUs process data in one direction only, which may miss important backward dependencies in text data [20].

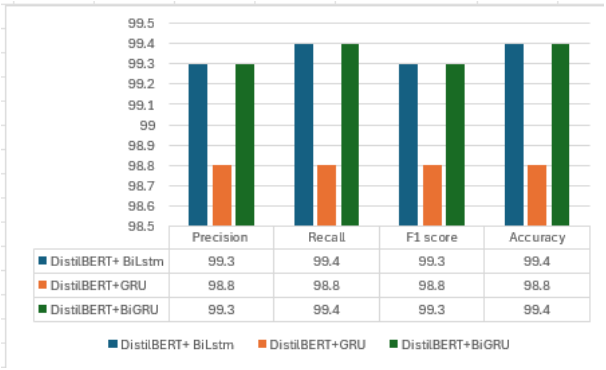

Fig. 7. Performance levels of Deep learning models with Distil bert in terms of Accuracy, Precision, Recall, and F1 score.

From Figure 7, we can clearly see that DistilBERT + BiLSTM and DistilBERT + BiGRU deliver the best results, with Precision, Recall, F1 Score, and Accuracy all achieving 99.3% to 99.45%. DistilBERT + GRU shows the lowest performance in this category, with all scores around 98.8%. GRUs process data in one direction only, which may miss important backward dependencies in text data. Therefore, in this case, the performance level is comparatively lower.

Transformer models like DistilBERT and RoBERTa produce high-quality contextual embeddings. When these embeddings are fed into BiLSTM or BiGRU, the recurrent layers can fine-tune the sequence-level understanding, leading to improved performance [21].

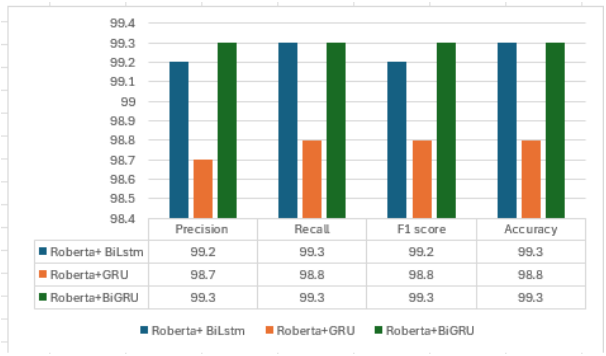

Fig. 8. Performance levels of Deep learning models with Roberta in terms of Accuracy, Precision, Recall, and F1 score.

Figure 8 demonstrates Roberta + BiGRU achieves the highest results across all metrics (99.3). RoBERTa + BiLSTM also performs well, scoring 99.2 to 99.3. BiGRU is computationally lighter than BiLSTM because it uses fewer gating mechanisms (no cell state), making training faster and less resource-intensive. RoBERTa + GRU shows the lowest performance for this group, with scores ranging between 98.7% and 98.8%.

1) Summary of the result of deep learning models with transformer::

- **Best Performing Models:**
  - DistilBERT + BiLSTM and DistilBERT + BiGRU consistently achieve the highest scores (99.3 to 99.4) across all metrics.
  - ALBERT + BiGRU also shows strong performance, with slightly lower scores compared to DistilBERT-based combinations.
- **Lowest Performing Models:**
  - GRU-based combinations for all three transformers (ALBERT, Distil BERT, Roberta) consistently deliver the lowest performance, with scores ranging between 98.7 and 99.0.
- **General Trends:**
  - BiGRU and BiLSTM combinations outperform GRU combinations across all transformers.
  - DistilBERT tends to perform slightly better compared to ALBERT and RoBERTa when combined with BiLSTM and BiGRU.

Their framework			Ourframework		
Classifier	F1 score	Accuracy	Classifier	F1 score	Accuracy
BERT+ XGBClassifier	98	98	DistilBERT+ BiLstm	99.3	99.4
BERT+ BaggingClassifier	98	98	AlBERT+GRU	98.9	99
BERT+DecisionTreeClassifie	98	98	DistilBERT+BiGRU	99.3	99.4

Fig. 9. Performance level between our framework (deep learning with advanced Bert variants) and previous framework(machine learning with BERT)

2) Comparison between previous framework and our framework in domain classification:: From Figure 9, we can say our RQ3 is satisfied here. In the previous framework, Ferrara et al. got the highest 98% accuracy and f1 score with their best combinations (Bert with XGB, Bagging, and Decision Tree Classifier). But in our proposed framework, we successfully got the better result. They only work on assessing the f1 score and accuracy, but we also reveal the precision and recall score. For facilitating the comparison, we use here similar evaluation metrics. We also compare them with our best combinations. With Distil Bert+ BiLSTM and Distil Bert+BiGRU. we get the best 99.3%f1 score and 99.4% accuracy score where they only get 98% accuracy and f1 score. We get the highest performance level with GRU when we employ the Albert transformer. AlBert+GRU gives 98.9% fa score and 99 % accuracy which is better than their framework(98%). Therefore, we can enhance the performance level in classifying domains from user stories by applying deep learning techniques and answering the RQ3.

Results From ML Task Classification

In this section, we describe how Research Questions 2 (RQ2) and 4 (RQ4) were addressed through our experiments and evaluations. we explored traditional machine learning (ML) classification techniques using embeddings generated by transformer models like ALBERT, RoBERTa, and DistilBERT. These embeddings were used as input to classical classifiers, including Support Vector Machines (SVM), Random Forests, and Logistic Regression, to assess their effectiveness on domain-specific tasks. This hybrid approach revealed complementary strengths of deep learning models and traditional

ML techniques, providing a broader understanding of their performance.

A. Evaluating BERT Variants with Traditional ML Classification Techniques

To address RQ2—Do BERT-based models improve the classification of ML-specific tasks from user stories?—we experimented with various combinations of transformer-based embeddings and traditional ML classifiers.

Our findings show that the RoBERTa embeddings combined with Label Powerset and Decision Tree emerged as the most effective approach, achieving an F1 score of 0.89 and a Hamming loss of 0.04, as shown in Fig 10. This approach significantly outperformed prior studies, which reported an F1 score of 0.86 and a Hamming loss of 0.07 using BERT embeddings with Decision Trees, as seen in Fig 11. The improved performance highlights the potential of transformer models, particularly when paired with the right classification techniques, to substantially enhance the classification of ML-specific tasks from user stories. This demonstrates that BERT-based models, when combined with appropriate classification methods, can effectively improve the task classification, thus satisfying RQ2.

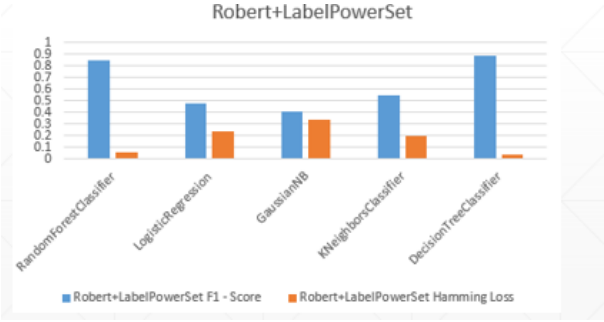

Fig. 10. Performance of Traditional ML algorithms with Bert Version Robert

Current Framework

BERT		
Technique + Model	F1-Score	Hamming Loss
LP + DT	0.86	0.07
LP + RF	0.84	0.08
BR + DT	0.78	0.11

Our Framework

Robert+LabelPowerSet		
ML Model	F1 - Score	Hamming Loss
RandomForestClassifier	0.85	0.06
LogisticRegression	0.48	0.24
GaussianNB	0.41	0.34
KNeighborsClassifier	0.55	0.2
DecisionTreeClassifier	0.89	0.04

Fig. 11. Comparision between their Framework and our framework with ML classifier

B. Evaluating BERT Variants with Deep Learning Model for ML Task Classification

To answer RQ4, we evaluated various deep learning models incorporating transformer-based embeddings and recurrent architectures to classify ML-specific tasks effectively. Our analysis focused on performance metrics such as Precision, Recall, F1 Score, and Hamming Loss, showcasing the effectiveness of each model.

The RoBERTa with GRU model demonstrated exceptional performance, achieving a Precision of 0.9976, a Recall of 0.9987, and F1 Score of 0.9981, and a Hamming Loss of 0.0009. F1 Score for BiGRU was also very high of .996 with very low Hamming Loss .0091. This model excelled in precision and recall, reflecting its ability to classify multi-label tasks effectively as shown in fig 12.

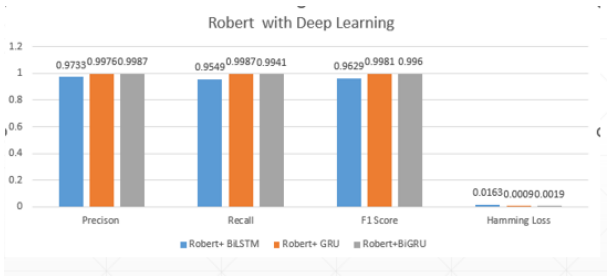

Fig. 12. Performance level of Robert With GRU and BiGRU

The ALBERT with GRU model achieved the highest overall performance, with a Precision of 0.9991, a Recall of 0.9987, an F1 Score of 0.9989, and a Hamming Loss of 0.0005. This model stood out due to its minimal classification errors, indicated by the lowest Hamming Loss among all models tested as shown in fig 13.

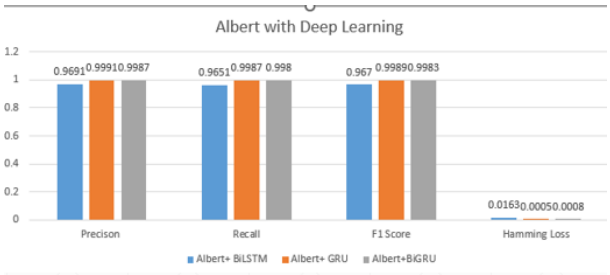

Fig. 13. Performance level of Albert With GRU and BiGRU and BiLSTM

Similarly, the ALBERT with BiGRU model performed comparably, achieving a Precision of 0.9987, a Recall of 0.9980, an F1 Score of 0.9983, and a Hamming Loss of 0.0008. These results underscore the robustness of ALBERT embeddings when paired with GRU-based architectures.

In contrast, while models such as DistilBERT with BiLSTM, DistilBERT with GRU, and DistilBERT with BiGRU achieved F1 Scores in the range of 0.9465 to 0.9475 and Hamming Loss values around 0.0245 to 0.0255, they did not match the superior performance of the ALBERT and RoBERTa-based models.

Thus, by combining the power of deep learning and context-aware word embeddings, we have achieved RQ4, demonstrating that models such as ALBERT with GRU and ALBERT with BiGRU performed exceptionally well in detecting and classifying machine learning tasks. Specifically, the ALBERT with GRU model achieved the highest performance with an

F1 Score of 0.9989 and a Hamming Loss of 0.0005, closely followed by ALBERT with BiGRU, which achieved an F1 Score of 0.9983 and a Hamming Loss of 0.0008. When compared with current studies fig 14, these models have surpassed previous benchmarks, achieving higher F1 scores and lower Hamming Loss, thereby setting a new standard in ML task detection. These results highlight the capability of these deep learning models to enhance task detection accuracy during the early stages of software development, addressing fairness concerns and laying a solid foundation for future improvements in requirements engineering.

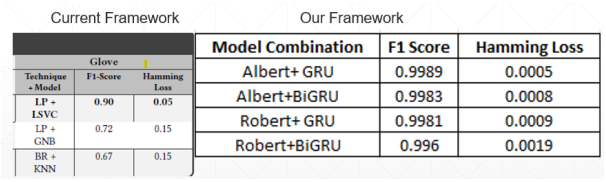

Fig. 14. Comparison between our framework and existing framework with Deep Learning Model

C. Top Performers

The following table 1 highlights the F1 Score and Hamming Loss for all model combinations, with the top performing models highlighted. Additionally,fig 15 presents a heatmap showing the accurate identification of all labels by the top-performing model Albert with GRU.

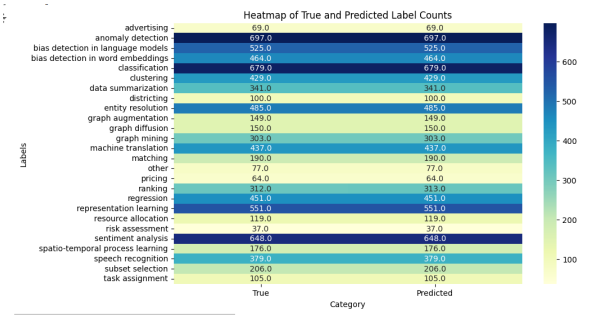

Fig. 15. HeatMap Albert with GRU

Model Combination	F1 Score	Hamming Loss
ALBERT with GRU	0.9989	0.0005
ALBERT with BiGRU	0.9983	0.0008
RoBERTa with GRU	0.9981	0.0009
RoBERTa with BiLSTM	0.9629	0.0163
DistilBERT with BiLSTM	0.9475	0.0245
DistilBERT with GRU	0.9475	0.0245
DistilBERT with BiGRU	0.9465	0.0255
ALBERT with BiLSTM	0.9670	0.0163

TABLE I
EVALUATION METRICS FOR ALL MODEL COMBINATIONS. F1 SCORE AND HAMMING LOSS ARE SHOWN.

VI. CONCLUSION

In this work, we extended the methodology proposed by ReFair, which focuses on ensuring fairness in machine learning predictions during the requirements engineering phase. The ReFair framework provided a context-aware approach for classifying sensitive features in User Stories using natural language processing and word embedding techniques. However, while ReFair demonstrated high accuracy in identifying sensitive features and domain-specific tasks, it was limited by the performance of traditional models.

In conclusion, our study addressed the limitations of traditional machine learning approaches by incorporating deep learning models and advanced word embedding techniques. This approach resulted in significant improvements in both multi-class classification for domain detection and multi-label classification for ML task detection. For domain detection, which involved multi-class classification, the DistilBERT with BiLSTM and DistilBERT with BiGRU models achieved impressive results, with an F1 score of 0.993 and 99.4% accuracy, demonstrating their effectiveness in this task. For ML task detection, a multi-label classification problem, the ALBERT with GRU model led to the highest performance, with an F1 score of 0.9989 and a Hamming loss of 0.0005. These results highlight the efficacy of deep learning models

VII. FUTURE WORK

While this work has shown promising results in enhancing machine learning task detection and domain classification during the early stages of requirements gathering, there are several avenues for future research. One key area for improvement is the use of a more diverse and original set of datasets. The current study utilized a synthetic dataset, which, while useful for demonstrating the capabilities of the framework, may not fully capture the complexity and variety of real-world user stories. Using authentic and diverse user stories from different domains will help validate and refine the models further, ensuring they can handle a wider range of real-world scenarios.

Lastly, expanding the scope of the framework to handle more complex, multi-modal requirements (e.g., including user feedback or domain-specific constraints) would further enhance its practical applicability. Future work should also explore the possibility of incorporating fairness-aware techniques more comprehensively throughout the entire requirements engineering process, enabling more equitable machine learning solutions.

REFERENCES

[1] C. Ferrara, F. Casillo, C. Gravino, A. De Lucia, and F. Palomba, "Refair: Toward a context-aware recommender for fairness requirements engineering," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–12.

[2] A. Riel, "Object oriented design heuristics/arthur j. riel," 1996.

[3] A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla, "Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary," *Journal of artificial intelligence research*, vol. 61, pp. 863–905, 2018.

[4] S. Azimov, "Paraphrasing user stories with large language models," 2024.

[5] G. B. Herwanto, G. Quirchmayr, and A. M. Tjoa, "Leveraging nlp techniques for privacy requirements engineering in user stories," *IEEE Access*, 2024.

[6] J. Kääriäinen, J. Koskela, J. Takalo, P. Abrahamsson, and K. Kolehmainen, "Supporting requirements engineering in extreme programming: managing user stories," in *Proc. 16th International Conference on Software & Systems Engineering and their Applications (ICSSEA 2003)*. Citeseer, 2003.

[7] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper, "Improving agile requirements: the quality user story framework and tool," *Requirements engineering*, vol. 21, pp. 383–403, 2016.

[8] I. K. Raharjana, D. Siahaan, and C. Fatichah, "User story extraction from online news for software requirements elicitation: A conceptual model," in *2019 16th International Joint Conference on computer science and Software Engineering (JCSSE)*. IEEE, 2019, pp. 342–347.

[9] Z. Lan, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[10] A. Areshey and H. Mathkour, "Exploring transformer models for sentiment classification: A comparison of bert, roberta, albert, distilbert, and xlnet," *Expert Systems*, vol. 41, no. 11, p. e13701, 2024.

[11] V. Dogra, A. Singh, S. Verma, Kavita, N. Jhanjhi, and M. Talib, "Analyzing distilbert for sentiment classification of banking financial news," in *Intelligent Computing and Innovation on Data Science: Proceedings of ICTIDS 2021*. Springer, 2021, pp. 501–510.

[12] G. Xu, Y. Meng, X. Qiu, Z. Yu, and X. Wu, "Sentiment analysis of comment texts based on bilstm," *Ieee Access*, vol. 7, pp. 51 522–51 532, 2019.

[13] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE, 2017, pp. 1597–1600.

[14] Y. Duan, H. Li, M. He, and D. Zhao, "A bigru autoencoder remaining useful life prediction scheme with attention mechanism and skip connection," *IEEE Sensors Journal*, vol. 21, no. 9, pp. 10 905–10 914, 2021.

[15] N. Kyurkchiev and S. Markov, "Sigmoid functions: some approximation and modelling aspects," *LAP LAMBERT Academic Publishing, Saarbrucken*, vol. 4, 2015.

[16] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," in *International conference on Machine learning*. PMLR, 2023, pp. 23 803–23 828.

[17] S. Bock and M. Weiß, "A proof of local convergence for the adam optimizer," in *2019 international joint conference on neural networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[18] H. Kaleem, S. Liaqat, M. T. Hassan, A. Mehmood, U. Ahmad, and A. Ditta, "An intelligent healthcare system for detecting diabetes using machine learning algorithms," *Lahore Garrison University Research Journal of Computer Science and Information Technology*, vol. 6, no. 03, pp. 1–11, 2022.

[19] S. Singla and N. Ramachandra, "Comparative analysis of transformer based pre-trained nlp models," *Int. J. Comput. Sci. Eng*, vol. 8, pp. 40–44, 2020.

[20] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE, 2017, pp. 1597–1600.

[21] A. F. Adoma, N.-M. Henry, and W. Chen, "Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition," in *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. IEEE, 2020, pp. 117–121.