Ensuring Test Case Coverage through Multilabel Requirements Classification: A Feasibility Study with Pre-Trained Models

Flávia Oliveira*
Sidia Institute of Science and Technology
Manaus, Brazil
flavia.oliveira@sidia.com

Ana Paula Silva*
Sidia Institute of Science and Technology
Manaus, Brazil
silva.ana@sidia.com

Leonardo Tiago
Sidia Institute of Science and Technology
Manaus, Brazil
leonardo.albuquerque@sidia.com

Lennon Chaves
Sidia Institute of Science and Technology
Manaus, Brazil
lennon.chaves@sidia.com

ABSTRACT

Context: In a software development project, the developers' main task is to understand the requirements to then implement them without bugs. Particularly, this paper is inserted in the context of a test team from a software institute that receives new requirements daily. With the goal of verifying if the requirements were correctly implemented, test cases are created based on these requirements, to then confirm if the software is working. **Problem**: Each requirement must be associated to one or more test cases, thus allowing tracking and ensuring the validation during the test process. However, manually classifying requirements to their associated test cases demands time and effort. **Solution**: This work details a study about the viability of using the pre-trained models: BERT, Electra, RoBERTa, DeBERTa and XLNet in the process of multi-label classification of requirements, identifying the test cases associated to each requirement. **Method**: For this study, we considered requirements received by the test team from 2024 to April 2025, and 38 associated test cases. After collecting and preparing the data, we trained the models and measured their performance through the metrics: Hamming Loss, Jaccard, Precision, Recall and F1-Score. **Results**: Results showed that all models had low Hamming Loss values, specially the BERT model, with 0.012 Hamming Loss. After performing the Friedman and Conover tests, it was determined that there is a significative discrepany between the models. **Conclusions**: Therefore, this research's results show that it is possible to use pre-trained models to classify requirements with associated test cases.

KEYWORDS

Software Test, Test Case, Pre-Trained Model, Requirement Classification

1 Introduction

The software development life cycle (SDLC) starts before technical activities and goes on until the final stages of the project. It is a structured process that aims to guarantee software quality [22]. SDLC models follow common steps, such as: requirements elicitation, analysis, design, development, testing, implementation and maintenance [4]. Among these steps, requirements elicitation is essential, as they define the goals, features and restrictions that the system must follow [30]. When we ensure that the needs are being correctly met from the start, we can avoid errors, reduce rework and ease the management of changes along the project [30].

To ensure that the requirement was correctly implemented in the software, testers must perform the tests as specified in test cases [54]. This way, test cases are written based on the Specification of Software Requirements, which contain all the functional and non-functional requirements of the software [35].

A test case is composed of preconditions, reproduction steps and expected results, with the goal of verifying the software's behavior and identifying the highest number of issues in a short time, thus minimizing costs and ensuring product quality [14, 35]. The test cases' coverage of requirements involves traceability, which ensures that tests are aligned to the requirements, and validation, that ensures that tests are effectively detecting errors [54]. However, since a requirement can change during the project [30], it is essential to ensure that the test case follows the change in requirement [30].

Thus, as new requirements are implemented, keeping the test cases up to date is an essential process to ensure the quality and efficacy of the process of a system [25]. Requirements are frequently updated as the client desires during a SDLC, and as a consequence, test cases may or may not be impacted by the changes. Keeping the test case outdated can result in inconsistencies, issues not being detected and rework [13]. This way, the adequate coverage of requirements through test cases ensures that each feature desired by the client is being validated [2].

Considering this scenario, it is also important to reflect on how requirements are initially collected and organized. Manually collecting and classifying requirements demands effort and is prone to error [39]. With the goal of minimizing the time spent and possible errors, an alternative is to use pre-trained models for analysing requirements [27], due to their capacity of understand, generate and process text [27].

In a software test team that performs functional tests and receives a great amount of requirements, analysing and classifying them according to the associated test cases demands time and conflicts with other daily activities performed by the team. Also, since the amount of requirements is large, there is the possibility of them being wrongly classified. During classification, it is necessary to verify if there is more than one test case associated with the requirement, making this process a Multi-Label Classification [11].

*Both authors contributed equally in this research.

This way, this paper proposes an approach based on pre-trained models to automatically determine which test cases are impacted by changes in the requirements within a test team in a software institute. The test team in which this study was performed is responsible for performing functional tests during the software development stage, and receives requirements daily. Therefore, the members of the test team must link the associated test cases to a requirement, as associating requirements to many test cases not only allows for a more comprehensive validation, but also helps to identify quickly which tests must be revised or updated given the changes [55].

To perform this research, we considered the requirements received from 2024 to April 2025, and used the pre-trained models BERT [12], ELECTRA [8], RoBERTa [26], DeBERTa [16] and XLNet [49] with the goal of evaluating how each of them performs when classifying test cases according to the description of their associated requirement, and thus verifying the viability of using these models for this activity. This way, as a contribution for the industry, this approach has the potential of enhancing the traceability between requirements and tests, and reduce rework, offering an efficient solution for the software engineering field.

The remainder of the article is divided in the following manner: Section 2 presents the theoretical basis for this research; Section 3 details the process of classifying requirements; Section 4 discusses the methodology employed in this work; Section 5 goes over the results obtained in the training and test steps of the models; Section 6 presents discussions over the conducted a study; Section 7 states the limitations; and lastly, Section 8 discusses our conclusions regarding this work.

2 Theoretical Reference

2.1 Pre-Trained Models

Pre-trained models emerged from the hurdles faced by deep neural networks which were limited in contexts where data was scarce [15]. This occurs mainly because deep neural networks possess a great number of parameters, which tend to overfit when there are not sufficient data for training. As consequence, their capacity of generalization is compromised [15].

The term "pre-train" is related to the wider concept of Transfer Learning [42, 43], which emerged as a solution for these hurdles, defining the idea of using previously acquired knowledge to deal with new problems [15, 42]. This approach is divided in two phases: pre-training and fine-tuning [15]. In pre-training, the model acquires knowledge from great volumes of unlabeled data, which is later transferred, during fine-tuning, to the target tasks using labeled data [15]. With this approach, based in pre-training and fine tuning, the pre-trained models became a popular option for Natural Language Processing tasks, representing a major advancement for the field [15, 42].

Before the introduction of the Transformer architecture [40], the advancements in language processing were focused in fixed vector representations of words, such as Word2Vec [28] and GloVe [31]. These models improved the performance of "downstream" tasks, such as recognition of named entities [31], marking of grammatical classes [9] and answering questions [46].

Transformer is an architecture based on the attention mechanism, in which a "weight" is attributed to the more relevant parts of the input when processing each word. It is structured in two parts: the Encoder, which reads and understands the input, and the Decoder, which generates the exit based on what the Encoder understood [40]. Through Transformers, it became possible to develop more powerful models, such as BERT (Encoder) and GPT (Decoder) [15, 42]. Later, new models were created to enhance the efficiency of text comprehension focused in the Encoder structure, such as RoBERTa [26], Electra [8] and XLNet [49].

2.2 Related Works

Many works have approached the use of techniques based on Machine Learning and Language Models applied to software engineering, with emphasis on requirements and tests, as proposed in the work by Derya et al. [23]. They used an approach based on a pre-trained model to automatically classify software requirements in three categories: type, priority and gravity. The results obtained shows that the model they used, DistilBERT [32], displayed a good performance for the text classification text, with a F1-Score value of 71%.

The work of Verma et al. [41] compares the performances of classification algorithms for recommending tags for questions and answers about software engineering in an online community. As a result, LinearSVM [17, 24] displayed better results in the task of classifying tags in comparison to the Random Forest [6], Logistic Regression [5] and Ridge Classifier algorithms.

Similarly, Alhaizaey et al. [1] present a study that evaluates pre-trained models for automatically identification and classification of non-functional requirements. In the study, the models BERT, RoBERTa, XLNet and DistilBERT were considered, and among these models, BERT stood out, with an accuracy value of 93.41%, showing that the use of pre-trained models is an alternative to reduce manual effort in the process of identifying and classifying non-functional requirements.

However, differently from traditional classification, in which each piece of data is associated to a single label, in Multi-Label Classification each piece of data is associated to a subset of labels, which can be correlated [50, 53]. Multi-Label Classification has been applied in text classification, as can be seen in the work by Xun et al.[48], in which the authors showed significant improvements on developing a new architecture of Correlation Networks (CorNet) [48] for the task of the Extreme Multi-Label Text Classification (XMTC). XMTC has applications in recommendation systems and auto-labeling of web documents [47].

This way, the work by Winkler et al. [44] presents an automatic approach to classify natural language requirements using convolutional neural networks (CNNs). The CNNs are used to implement a multi-label requirement classifier that assigns the probability of a requirement being classified to one or more methods considered for verification which assigns the probability of a requirement being classified to one or more methods considered for requirement verification within the institute where the study was developed.

For test case classification, the work by Zhao et al. [52] proposes an approach to classify test cases using a classifier based in Bidirectional Long Short-Term Memory (BiLSTM) [21]. Their approach involves classifying test cases efficiently to help developers, considering the existing problem of inadequate and unbalanced data.

The results showed that the approach helps developers to select test cases faster.

In these works, the use of classification algorithms leads to a reduction of manual effort through the automation of the process, with satisfactory results. This way, this work describes a viability study regarding the use of pre-trained models for automatically classifying requirements to the associated test cases as a multi-label classification problem, given that each requirement may be assigned to more than one test case. For this study, the following 5 pre-trained models were considered: BERT, Electra, RoBERTa, DeBERTa e XLNet.

3 Research Problem

In the test team, test cases are updated or created as new requirements emerge, through Jira[1] tasks. To analyse requirements and update test cases, there is a group within the test team responsible for analysing the new requirements and identify if it is applicable to the test scopes covered by the team. There is also another group responsible for managing test cases who, in the event of a requirement impacting the test scopes, must reformulate the test case or create a new one, ensuring that the new requirement is covered by the test case and is verified during test execution.

There are a few possibilities for requirements classification with test cases:

- **Test Case ID:** If the requirement is related to any of the test team's scopes, the tester must check if there are any test cases which correspond to the requirement. If there are one or more applicable test cases, the tester must indicate in the Jira task which test cases are impacted among the 38 test cases. The test case ID is represented by a 'T' followed by the identification number, a number which is assigned to each test case incrementally as they are created. It is important to highlight that there are test cases that became obsolete because their requirements are not applicable anymore, or they are from the Charter scope, or from other teams' scopes.
- **NEW:** If there are no corresponding test cases, it must be indicated in the task that no test cases cover the requirement, and a new test case must be created.
- **NOT:** This classification is adopted if the requirement does not correspond to any of the scopes handled by the test team.

However, this classification process is performed manually by members of the test team, demanding much time and effort. This way, this research proposes the use of pre-trained models to classify the requirements by automatically identifying the associated test cases.

In the next Section, we present the methodology used to develop this study, detailing the process followed to classify the requirements, the metrics used for validation and the planning of the study.

4 Research Methodology

To verify the viability of the use of pre-trained models to classify requirements with test cases, in this Section we present the necessary steps for conducting the study, described in Figure 1.

[1]https://www.atlassian.com/software/jira

As shown in Figure 1, the steps for conducting the study were the creation of the dataset, composed of requirements and associated test cases, followed by data processing, implementation of the pre-trained models, followed by the evaluation of the results, and the planning of the experimental study.

4.1 Dataset

For this experiment, we collected requirements from 2024 to April 2025 to compose the dataset, then we processed the data. The dataset consists of 591 requirements and 38 associated test cases, in which each test case corresponds to a class. It is important to highlight that each requirement can be related to one or more test cases, categorizing the problem as a multi-label classification problem [51].

4.2 Data Processing

After collecting the data, we processed the data in the dataset, following the steps below for the samples (requirements):

(1) Removal of stop-words: In general, stop-words are words that provide a structure for the text, but individually they are not meaningful for the text analysis [36]. Thus, the stop-words were identified and removed.
(2) Removal of special characters: In this step, we removed the special characters, such as punctuation, double spaces and symbols.
(3) Application of the stemming technique: The technique consists in reducing the word to its root, eliminating prefixes and suffixes. This way, stemming was applied in all the words from the requirements' descriptions [20].

The pieces of data were initially separated in 20% for the test set, and 80% for the training and validation sets.

4.3 Implementation

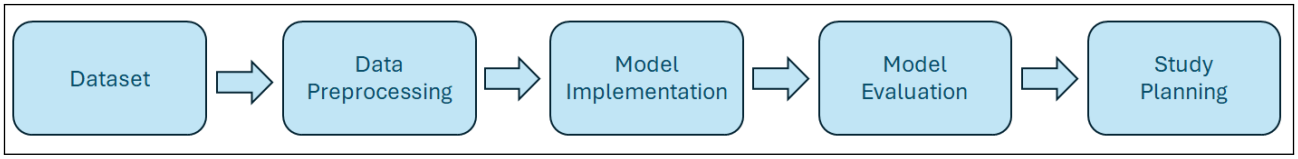
The implementation was made using Python 3.12.4[2] and a few of its libraries, such as Pandas, NumPy and SKLearn. The models BERT[3], Electra[4], XLNet[5], DeBERTa[6] and Roberta[7] were trained using a GPU server for 100 epochs with a batch size of 5. Furthermore, the Early Stopping callback mechanism was used, with a patience of 5 epochs, to interrupt the training in case the validation did not display improvements, thus avoiding overfitting the model.

To determine the viability of the use of pre-trained models in the task of classifying requirements, initially we trained the models with a dataset containing 473 requirements. Then, a dataset with 118 requirements was used for testing.

The models were chosen for the study because they represented different generations of language models. BERT was the first to popularize training with mask [12]; RoBERTa is an optimized version of BERT, with more data and adjustments [26]; ELECTRA proposes an approach based in the most efficient pre-training task in regards

[2]https://www.python.org/
[3]https://huggingface.co/google-bert/bert-base-uncased
[4]https://huggingface.co/google/electra-base-discriminator
[5]https://huggingface.co/xlnet/xlnet-base-cased
[6]https://huggingface.co/microsoft/deberta-v3-base
[7]https://huggingface.co/FacebookAI/roberta-base

Figure 1: Methodology applied to develop the study



to samples, called Replaced Token Detection; DeBERTA brought advancements to attention and embeddings [16]; and XLNet presents a different proposal from BERT's, as it considers different word orders [49].

4.4 Evaluation Metrics

To evaluate the performance of the models during the training and test steps, we used the evaluation metrics Hamming Loss, Jaccard, Precision, Recall and F1-Score. It is important to highlight that requirements can be classified with one, more than one or no test cases, thus categorizing this as a multi-label classification, and in this case, with the goal of measuring the development of the models, the Hamming Loss and Jaccard metrics were chosen.

4.4.1 Hamming Loss. The Hamming Loss [29] is the most used metric in multi-label classifications, calculating the distance between the real value and the predicted value for each piece of data. It is calculated through Equation 1 with $L_H(y, h(x))$, in which m represents the number of labels from the test dataset, $h(x)$ represents the label predicted by the classifier and y represents the true labels. Hamming Loss determines the proportion of labels that were incorrectly classified. This way, a lower score in this metric indicates a better effectiveness for the model.

$$L_H(y, h(x)) = \frac{1}{m} \sum_{i=1}^{m} y_i \neq h_i(x) \tag{1}$$

4.4.2 Jaccard. The Jaccard metric [38] is used to measure the similarities between sets of real and predicted labels. The higher the value of Jaccard similarity, the higher the correlation between the labels. The metric is calculated as illustrated in Equation 2, in which $|LS_i \cap LS_j|$ represents the number of identical labels in sets LS_i and LS_j, while $|LS_i \cup LS_j|$ represents the total number of labels. For this metric, a value closer to 1 indicates a higher similarity.

$$J(LS_i, LS_j) = \frac{|LS_i \cap LS_j|}{|LS_i \cup LS_j|} \tag{2}$$

4.4.3 F1-Score, Recall and Precision. The F1-Score metric includes Precision and Recall [33]. While the precision quantifies the positive predictions of the model and is represented by P in Equation 3, Recall measures the efficacy of the model in minimizing false-negatives and is represented by 'R' in the Equation 4, where TP represents true-positives and FP represents false-positives.

$$P = \frac{(TP)}{(TP + FP)} \tag{3}$$

$$R = \frac{(TP)}{(TP + FN)} \tag{4}$$

Lastly, the F1-Score metric is calculated as described in Equation 5, and is responsible for evaluating the balance between Recall and Precision, with the goal of identifying precisely the true-positives, minimizing false-positives and false-negatives.

$$\text{F1-Score} = \frac{(2xPxR)}{(P + R)} \tag{5}$$

4.5 Study Planning

To conduct the study on the viability of using pre-trained models to classify requirements, in this step were defined the scoping, the context, the hypothesis, and the selection of variables and subjects of the performed study with the goal of evaluating the models considering the performance of each one during the test step.

It is important to highlight that we do not use all the steps of the experimental process defined by Wohlin et al. [45]. Instead, we adapted it to bring scientific rigor to the study.

4.5.1 Scoping. The definition of this work's goal followed the GQM (Goal, Question, Metric) [3], described in Table 1:

Table 1: Goal of the study using the GQM pattern

Analyse	the performance of 5 different pre-trained models to classify requirements to the associated test cases
with the purpose of	evaluate
with respect to	probability distribution of correct classifications
from the point of view of	the researchers
in the context of	a software test team in a software institute

4.5.2 Context. The study was performed in a test team that acts in the software development phase in an institute that receives requirements daily, and must assign each requirement to one or more test cases, if applicable. The study was developed with models BERT, Electra, XLNet, DeBERTa and RoBERTa, and requirements to be classified with the associated test cases.

4.5.3 Hypothesis. The study was conducted with the goal of testing the following null and alternative hypothesis:

- H0: There is no difference between models BERT, Electra, XLNet, DeBERTa and RoBERTa in regards to the distribution of probabilities in test case classification.
- H1: There is a difference between models BERT, Electra, XLNet, DeBERTa and RoBERTa in regards to the distribution of probabilities in test case classification.

4.5.4 Variable Selections. The following variables were defined for the study:

- Independent variable: The training of the pre-trained models BERT, Electra, XLNet, DeBERTa e RoBERTa.
- Dependent variable: The probabilities of correct guesses in the classification of test cases for each model.

4.5.5 Selection of Subjects. The subjects of the study were the requirements, with each one associated to one or more of the 38 test cases, in which each test case corresponds to a class. The requirements are received daily by the test team and contain the following fields: Description (a brief summary of the requirement), Summary (description of what must be validated), Other Documents, and Test Case (the test case that will cover the requirement).

5 Results

In this Section, we present the results obtained through the fine-tuning of models BERT, Electra, XLNet, DeBERTa and RoBERTa in their respective training and test stages, following the process described in Section 4.

5.1 Training Results

For this viability study, Table 2 presents the result of the training metrics for each model, considering Precision, Recall, F1-Score, Jaccard and Hamming Loss, including the epoch in which each result was achieved.

As displayed in Table 2, the 5 models achieved a low Hamming Loss value, indicating that it is possible to use them to classify requirements with test cases. Results show that the BERT model presented a better Hamming Loss value, resulting in the lowest difference between the predicted and real labels, and a Jaccard value of 83%. BERT's performance also stands out compared to the other models used in the Precision (89%), Recall (85%) and F1-Score (87%) metrics.

However, BERT achieved these results after 27 epochs, while DeBERTa achieved a Hamming Loss value of 0.012 in the 18th epoch, but with a Jaccard value of 78%, a Precision value of 79%, a Recall value of 84% and an F1-Score value of 78%. When analysing the metrics achieved by RoBERTa, the Hamming Loss value achieved was 0.013 in the 17th epoch, a Jaccard value of 77%, a Precision value of 78%, a Recall value of 84% and a F1-Score value of 81%.

It is also noticed that the Electra model presented a Hamming Loss value of 0.011, with a Jaccard value of 77% and a value of 82% for the metrics Precision, Recall and F1-Score in the 26th epoch. The XLNet model, after 14 epochs, achieved a Hamming Loss value of 0.014, a Jaccard value of 75%, a Precision value of 84%, a Recall value of 72% and an F1-Score value of 77%.

5.2 Test Results

To measure the performance of the models, so as to verify the viability of their use in classifying requirements to the associated test cases, we present the results achieved by the 5 models in the test dataset.

As can be observed in Table 3, for the test dataset, all models achieved a low Hamming Loss value. Just as the results achieved during training, BERT achieved a lower Hamming Loss score than the rest, 0.012, and higher values for the other metrics, with a Jaccard value of 79%, an F1-Score value of 81%, a Recall value of 75% and a Precision value of 89%.

The RoBERTa, DeBERTa and XLNet models achieved a Hamming Loss value of 0.015. DeBERTa achieved Jaccard and Recall values of 76%, an F1-Score value of 79% and a Precision value of 82%. Meanwhile, XLNet achieved a Precision value of 88%, a Recall value of 69%, an F1-Score value of 77% and a Jaccard value of 75%. RoBERTa achieved a Precision value of 81%, a Recall value of 75%, an F1-Score value of 78% and a Jaccard value of 74%. Electra achieved a Hamming Loss value of 0.016, a Jaccard value of 73%, an F1-Score of 77%, a Recall value of 71% and a Precision value of 83%.

5.3 Overfitting Analysis

Figure 2 displays the overfitting curves for the 5 models. No model displays signs of overfitting, as all of them converged well, given that the training loss and validation loss are quickly decreased in the first epochs, which indicates that all models were able to learn how to do the task. The result seen in Figure 2 indicates that the BERT model achieved a lower loss value with both training and validation data. It is important to highlight that Early Stopping was used, which is why some models such as XLNet and RoBERTa are illustrated with fewer epochs when compared to BERT and Electra.

5.4 Statistical Analysis

To illustrate each model's probability distribution, we generated boxplots of the probability distribution of each model correctly predicting the test case. The boxplots are illustrated in Figure 3.

As can be seen in Figure 3, the BERT model displays a higher probability of success, despite the presence of outliers, with a median closer to that of models Electra and DeBERTa. It can also be seen that XLNet has more varied probability values, and a lower median value than BERT, Electra and DeBERTa. Furthermore, despite possessing outliers, RoBERTa displayed an even boxplot.

However, it is not possible to draw conclusions only by observing the boxplots. Thus, we performed statistical tests considering the probability distributions of correct classification of requirements, so as to determine if there is any significant difference between the five models.

Initially, a normality test was performed to check if the data follows a normal distribution. The test we chose was Shapiro Wilk [34], and the results were p value < 0.001, indicating that the samples do not follow a normal distribution.

Then, we chose the Friedman test [56], considering a $\alpha = 0.05$ significance level to determine if there is a significant difference between the models when classifying requirements with test cases. The Friedman test was applied for the 5 models together, and achieved p value = 0.00000, indicating that there is a significant difference between the models. Thus, after the Friedman test, we applied the Conover test [10] pair by pair, to determine which of the models displays a significant difference. Holm's method [18] was also used to control the error rate when performing multiple comparisons. The result of the Conover test is displayed in Table 4.

With the results seen in Table 4, a statistical difference can be noticed between the 5 models. BERT and Electra achieved p value= 0.04807, while BERT and DeBERTa achieved p value= 0.00612, and

Table 2: Training Metrics Achieved by the Models

Model	Precision	Recall	F1 Score	Jaccard	Hamming Loss	Epoch
BERT	89%	85%	87%	83%	0.008	27
RoBERTa	78%	84%	81%	77%	0.013	17
DeBERTa	79%	84%	81%	78%	0.012	18
XLNet	84%	72%	77%	75%	0.014	14
Electra	82%	82%	82%	77%	0.011	26

Table 3: Test Metrics Achieved by the Models

Model	Precision	Recall	F1 Score	Jaccard	Hamming Loss
BERT	89%	75%	81%	79%	0.012
RoBERTa	81%	75%	78%	74%	0.015
DeBERTa	82%	76%	79%	76%	0.015
XLNet	88%	69%	77%	75%	0.015
Electra	83%	71%	77%	73%	0.016

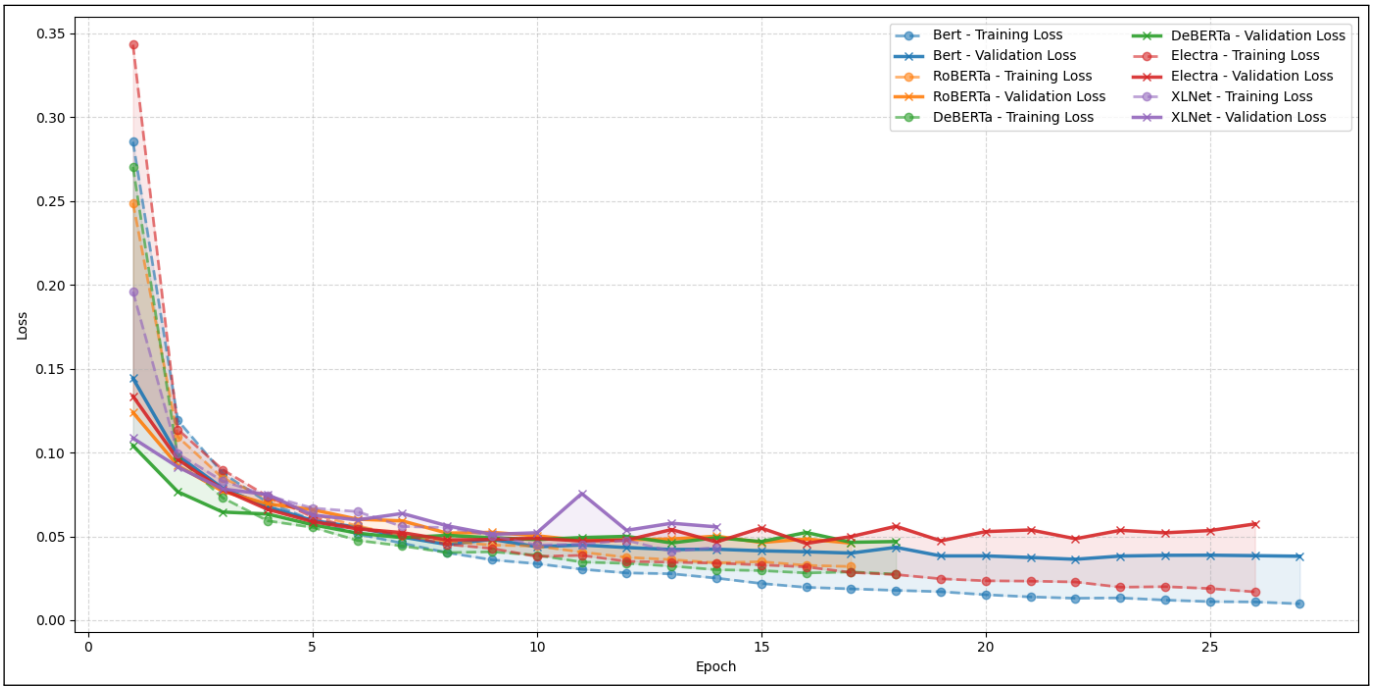


Figure 2: Overfitting Comparison Across Models

Table 4: Results of the Conover test

	BERT	Electra	XLNet	DeBERTa	RoBERTa
BERT	1.00000	0.04807	0.00000	0.00612	0.00000
Electra	0.04807	1.00000	0.00000	0.00000	0.00000
XLNet	0.00000	0.00000	1.00000	0.00000	0.00280
DeBERTa	0.00612	0.00000	0.00000	1.00000	0.00000
RoBERTa	0.00000	0.00000	0.00280	0.00000	1.00000

RoBERTa and XLNet achieved p value = 0.00280. The rest of the model pairs achieved p value = 0.0000.

Thus, for the problem of classifying requirements with test cases, the statistical tests show that the 5 models are different among

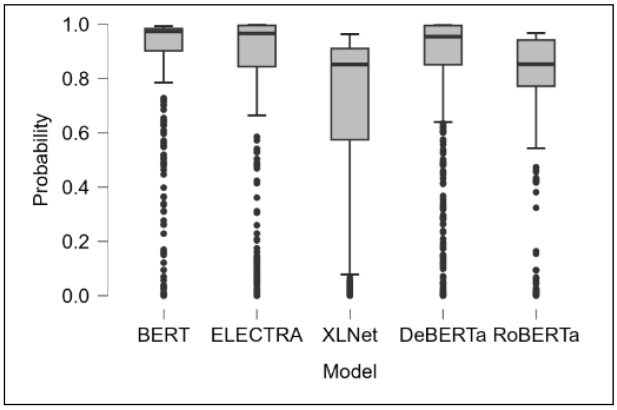


Figure 3: Boxplots of the probability distributions of correct classification of the requirement of each model

themselves, therefore all of them can be seen as an alternative for the multi-label classification problem, when associating or or more test cases to the requirement. BERT seems to be the best suited for this problem, for displaying a better performance than the rest of the models in the metrics used for evaluation: Hamming Loss, Jaccard, Precision, Recall and F1-Score. It also has a boxplot with the highest median value, indicating a higher probability of success in classifying requirements.

Therefore, with the presented results, it is possible to reject the null hypothesis H0, given that the statistical tests showed that there is a significant difference between the models, and based on the test metrics seen in Table 3, BERT achieved the best performance for the task of classifying requirements with test cases, considering the metrics used.

6 Discussions

This study presented the viability of using pre-trained models to classify requirements with test cases in a software test team, highlighting the use of this approach in a real scenario from the industry. The models used - BERT, RoBERTa, Electra, DeBERTa and XLNet - proved themselves to be capable of classifying requirements with test cases, and among them, BERT achieved the best results for the metrics used to evaluate the problem.

Works such as the ones by Ahmad et al. [37] e Derya et al. [23] also explore the use of BERT to classify requirements in different contexts. Chen et al. [7] present an empirical study to explore and improve the performance of multi-label text classification using different layers of a pre-trained BERT model. While Nuno et al. [27] investigated the use of ChatGPT to help with requirements engineering, Sayem et al. [19] used machine learning techniques to predict if a requirement is vulnerable for the system. These studies still focused in the classification or interpretation of requirements, without associating the test cases impacted by these new requirements. Therefore, performing this classification of requirements with test cases ensures that these requirements will be checked during software testing.

Differently of the previous approaches, that showed that pre-trained models can understand well the requirements' natural language, our approach displays that pre-trained models can act directly in the software engineering process as an alternative of automating the process of classifying requirements to the associated test cases.

Thus, we show in this study that it is viable to implement what has been trained in a real scenario from the industry, making the traceability between requirements and test cases automated and efficient. This way, it is possible to use pre-trained models to automate the task of classifying requirements by associating them to one or more related test cases, minimizing possible human error and speeding up the process, thus allowing for more time and resources to be assigned to other tasks.

7 Limitations of the Study

Some limitations must be considered for this study:

- The study was conducted in a software institute with confidentiality policies and a specific development context, thus the models and datasets cannot be shared.
- The models were trained using requirements and test cases specific to the study's test team of choice, and cannot be generalized to other teams of the same institute.
- The study was based on a dataset composed of 591 requirements and 38 associated test cases, so more data needs to be collected to improve the results of the models used for classification.

8 Conclusion

This study describes the use of pre-trained models to classify requirements to associated test cases, with the goal of helping a test team from a software institute that receives new requirements daily and has to manually classify them. This way, we present the results of a viability study on the application of 5 models to automatically classify requirements.

We conducted the study considering requirements received by a software test team in a real scenario from the industry, and the models were trained and measured using the metrics Jaccard, Hamming Loss, F1-Score, Precision and Recall. Among the models, BERT achieved the best results for the evaluation metrics, with the lowest Hamming Loss value among the models, and superior values for Jaccard, Precision, Recall and F1-Score. Then, we performed statistical tests to check if there is a significant difference among the models. The result of Friedman's test indicated that there is such a difference among the models, and the Conover test allowed us to determine that all the models are significantly different among themselves in regards to the probability of correctly classifying requirements with test cases.

With the presented results, it is possible to determine the viability of the use of pre-trained models to classify requirements with test cases, with emphasis on the BERT model, which achieved the best results for the considered metrics: a Hamming Loss value of 0.012, a Jaccard value of 79%, an F1-Score value of 81%, a Recall value of 75% and a Precision value of 89%.

For future works, we intend to expand the dataset with new requirements. We also intend to conduct a case study after the

implementation of the model to classify requirements with test cases within the test team, to determine the advantages and disadvantages of the automated process. While this work covered the identification of the requirement and of the associated test case, we intend to use Large Language Models (LLMs) to suggest the changes that must be made to a test case based on the new requirement, or to suggest the creation of a new test case.

ACKNOWLEDGMENTS

This paper is a result of the Research, Development & Innovation Project (ASTRO) performed at Sidia Institute of Science and Technology sponsored by Samsung Eletrônica da Amazônia Ltda., using resources under terms of Federal Law No. 8.387/1991, by having its disclosure and publicity in accordance with art. 39 of Decree No. 10.521/2020. The authors extend their gratitude to the Software Engineering Laboratory for its contribution to the IT infrastructure.

REFERENCES

[1] Abdulrahim Alhaizaey and Majed Al-Mashari. 2025. Automated Classification and Identification of Non-Functional Requirements in Agile-Based Requirements Using Pre-Trained Language Models. *IEEE Access* (2025).

[2] Osman Balci. 1998. Verification, validation, and testing. *Handbook of simulation* 10, 8 (1998), 335–393.

[3] Victor R Basili. 1994. Goal, question, metric paradigm. *Encyclopedia of software engineering* 1 (1994), 528–532.

[4] T Bhuvaneswari and S Prabaharan. 2013. A survey on software development life cycle models. *International Journal of Computer Science and Mobile Computing* 2, 5 (2013), 262–267.

[5] Dankmar Böhning. 1992. Multinomial logistic regression algorithm. *Annals of the institute of Statistical Mathematics* 44, 1 (1992), 197–200.

[6] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[7] Zhichao Chen, Han Liu, and Qin Zhang. 2023. Adaptive Selection of BERT Layer for Multi-Label Text Classification. In *2023 International Conference on Machine Learning and Cybernetics (ICMLC)*. IEEE, 434–439.

[8] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555* (2020).

[9] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. (2011).

[10] William Jay Conover. 1999. *Practical nonparametric statistics*. john wiley & sons.

[11] André CPLF de Carvalho and Alex A Freitas. 2009. A tutorial on multi-label classification techniques. *Foundations of Computational Intelligence Volume 5: Function Approximation and Classification* (2009), 177–195.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. doi:10.18653/v1/N19-1423

[13] Robert Feldt. 2014. Do System Test Cases Grow Old?. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation*. 343–352. doi:10.1109/ICST.2014.47

[14] Dorothy Graham, Erik van Veenendaal, Isabel Evans, and Rex Black. 2008. *Foundations of software testing: ISTQB certification*. Intl Thomson Business Pr.

[15] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. 2021. Pre-trained models: Past, present and future. *AI Open* 2 (2021), 225–250.

[16] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654* (2020).

[17] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their applications* 13, 4 (1998), 18–28.

[18] Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* (1979), 65–70.

[19] Sayem Imtiaz, Md Rayhan Amin, Anh Quoc Do, Stefano Iannucci, and Tanmay Bhowmik. 2021. Predicting vulnerability for requirements. In *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, 160–167.

[20] Abdul Jabbar, Sajid Iqbal, Manzoor Ilahi Tamimy, Amjad Rehman, Saeed Ali Bahaj, and Tanzila Saba. 2023. An analytical analysis of text stemming methodologies in information retrieval and natural language processing systems. *IEEE Access* 11 (2023), 133681–133702.

[21] Zahra Jamshidzadeh, Mohammad Ehteram, and Hanieh Shabanian. 2024. Bidirectional Long Short-Term Memory (BILSTM)-Support Vector Machine: A new machine learning model for predicting water quality parameters. *Ain Shams Engineering Journal* 15, 3 (2024), 102510.

[22] Pravin M Kamde, VD Nandavadekar, and RG Pawar. 2006. Value of test cases in software testing. In *2006 IEEE International Conference on Management of Innovation and Technology*, Vol. 2. IEEE, 668–672.

[23] Derya Kici, Garima Malik, Mucahit Cevik, Devang Parikh, and Ayse Basar. 2021. A BERT-based transfer learning approach to text classification on software requirements specifications.. In *Canadian AI*.

[24] Ja-Yong Koo, Yoonkyung Lee, Yuwon Kim, and Changyi Park. 2008. A Bahadur Representation of the Linear Support Vector Machine. *Journal of Machine Learning Research* 9, 44 (2008), 1343–1368. http://jmlr.org/papers/v9/koo08a.html

[25] Carlos DQ Lima, Everton LG Alves, and Wilkerson L Andrade. 2024. A Systematic Literature Review on MBT Test Cases Maintenance. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 1356–1365.

[26] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[27] Nuno Marques, Rodrigo Rocha Silva, and Jorge Bernardino. 2024. Using ChatGPT in Software Requirements Engineering: A Comprehensive Review. *Future Internet* 16, 6 (2024). doi:10.3390/fi16060180

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26 (2013).

[29] Mohamad Syahrul Mubarok, Nanang Saiful Huda, et al. 2018. A multi-label classification on topics of quranic verses in English translation using Tree Augmented Naïve Bayes. In *2018 6th International Conference on Information and Communication Technology (ICoICT)*. IEEE, 103–106.

[30] Bashar Nuseibeh and Steve Easterbrook. 2000. Requirements engineering: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering* (Limerick, Ireland) *(ICSE '00)*. Association for Computing Machinery, New York, NY, USA, 35–46. doi:10.1145/336512.336523

[31] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[32] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[33] Guzman Santafe, Iñaki Inza, and Jose A Lozano. 2015. Dealing with the evaluation of supervised classification algorithms. *Artificial Intelligence Review* 44 (2015), 467–508.

[34] Samuel Sanford Shapiro and Martin B Wilk. 1965. An analysis of variance test for normality (complete samples). *Biometrika* 52, 3-4 (1965), 591–611.

[35] Navnath Shete and Avinash Jadhav. 2014. An empirical study of test cases in software testing. In *International Conference on Information Communication and Embedded Systems (ICICES2014)*. IEEE, 1–5.

[36] Gurpej Singh, Rahul Bhandari, and Prabhdeep Singh. 2024. Advancing NLP for Punjabi Language: A Comprehensive Review of Language Processing Challenges and Opportunities. In *2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*. IEEE, 1250–1257.

[37] Ahmad F Subahi. 2023. Bert-based approach for greening software requirements engineering through non-functional requirements. *IEEE Access* 11 (2023), 103001–103013.

[38] Nor Hashimah Sulaiman and Daud Mohamad. 2012. A Jaccard-based similarity measure for soft sets. In *2012 IEEE symposium on humanities, science and engineering research*. IEEE, 659–663.

[39] Chetan Surana Rajender Kumar Surana, Dipesh B Gupta, Sahana P Shankar, et al. 2019. Intelligent chatbot for requirements elicitation and classification. In *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*. IEEE, 866–870.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[41] Ayushi Verma and Neetu Sardana. 2019. Comparative Study of Multilabel Classifiers on Software Engineering Q&A Community for Tag Recommendation. In *2019 International Conference on Signal Processing and Communication (ICSC)*. IEEE, 190–194.

[42] Haifeng Wang, Jiwei Li, Hua Wu, Eduard Hovy, and Yu Sun. 2023. Pre-trained language models and their applications. *Engineering* 25 (2023), 51–65.

[43] Xiao Wang, Guangyao Chen, Guangwu Qian, Pengcheng Gao, Xiao-Yong Wei, Yaowei Wang, Yonghong Tian, and Wen Gao. 2023. Large-scale multi-modal pre-trained models: A comprehensive survey. *Machine Intelligence Research* 20, 4

(2023), 447–482.
[44] Jonas Paul Winkler, Jannis Grönberg, and Andreas Vogelsang. 2019. Predicting how to test requirements: An automated approach. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 120–130.
[45] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. 2012. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.
[46] Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dcn+: Mixed objective and deep residual coattention for question answering. *arXiv preprint arXiv:1711.00106* (2017).
[47] Jie Xiong, Li Yu, Xi Niu, and Youfang Leng. 2023. XRR: Extreme multi-label text classification with candidate retrieving and deep ranking. *Information Sciences* 622 (2023), 115–132.
[48] Guangxu Xun, Kishlay Jha, Jianhui Sun, and Aidong Zhang. 2020. Correlation networks for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1074–1082.
[49] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).
[50] Tingting Zhai and Hao Wang. 2022. Online passive-aggressive multilabel classification algorithms. *IEEE transactions on neural networks and learning systems* 34, 12 (2022), 10116–10129.
[51] Min-Ling Zhang and Zhi-Hua Zhou. 2013. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering* 26, 8 (2013), 1819–1837.
[52] Yuan Zhao, Sining Liu, Quanjun Zhang, Xiuting Ge, and Jia Liu. 2023. Test case classification via few-shot learning. *Information and Software Technology* 160 (2023), 107228.
[53] Zhi-Hua Zhou and Min-Ling Zhang. 2017. Multi-label Learning.
[54] Hong Zhu, Patrick A. V. Hall, and John H. R. May. 1997. Software unit test coverage and adequacy. *ACM Comput. Surv.* 29, 4 (Dec. 1997), 366–427. doi:10.1145/267580.267590
[55] Celal Ziftci and Ingolf Krüger. 2013. Getting more from requirements traceability: Requirements testing progress. In *2013 7th international workshop on traceability in emerging forms of software engineering (TEFSE)*. IEEE, 12–18.
[56] Donald W Zimmerman and Bruno D Zumbo. 1993. Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks. *The Journal of Experimental Education* 62, 1 (1993), 75–86.