Handling Ambiguity in App Review-Based Software Requirement Classification Using Multi-Label BERT Transfer Learning

Stefani Tasya Hallatu [1,*)], Muhammad Jerino Gorter [2)], Andrea Bemantoro Jati [3)], Diana Purwitasari [4)], Chastine Fatichah [5)], and Hilya Tsaniya [6)]

[1, 2, 3, 4, 5, 6)] Department of Informatics, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia

E-mail: 6025241011@student.its.ac.id[1)], 6025241054@student.its.ac.id[2)], 6025241047@student.its.ac.id[3)], diana@if.its.ac.id[4)], chastine@if.its.ac.id[5)], and 7025222004@student.its.ac.id[6)]

ABSTRACT

User-generated reviews on mobile applications represent a valuable yet ambiguous resource for classifying software requirements, particularly when multiple aspects such as bugs, feature requests, and user experiences are embedded within a single review. Although prior studies have shown the potential of transformer-based and multi-label models in improving text classification accuracy and efficiency, explicit handling of semantic ambiguity in multi-aspect reviews has not been addressed. This study proposes a multi-label classification approach using BERT-based transfer learning to manage ambiguity in app reviews. Each review is manually annotated with one or more relevant requirement categories. Preprocessing involves text cleaning, normalization, and BERT tokenization to convert reviews into structured representations. The classification model categorizes reviews into four classes: bug reports, feature requests, user experiences, and ratings. Evaluation results demonstrate strong performance, with F1-scores of 0.96 for bug reports, 0.95 for feature requests, 0.97 for ratings, and 0.80 for user experiences, confirming the model's capability in capturing overlapping labels in ambiguous reviews. This approach offers a scalable and automated solution for extracting software requirements, enabling developers to better identify, categorize, and prioritize user needs from unstructured review data.

Keywords: Ambiguity handling, app review classification, BERT, multi-label learning, software requirements.

1. Introduction

The growing use of mobile applications has generated millions of reviews from users. These reviews contain various types of information, including bug reports, feature requests, user experiences, and assessments of overall application quality. As stated by [1], app reviews can serve as a highly valuable source of feedback for the development, maintenance, and evolution of applications, especially in meeting the demands of a competitive and dynamic digital market. Compared to traditional survey methods, app reviews provide data at a much larger and more structured scale, enabling a more accurate representation of real consumer experiences as application users [2]. With the large amount of information contained in these reviews, various important insights can be obtained provided that the data is processed appropriately. As demonstrated in [3], applying K-Means clustering to user reviews can reveal sentiment patterns that provide meaningful input for developers in planning application improvements. Furthermore, app reviews can also be used to predict software evolution needs through techniques such as review classification, sentiment analysis, and utility prediction, enabling developers to design application roadmaps based on more objective and analytical information [4]. Other studies show that both functional and non-functional requirements can be indirectly inferred from user input, helping developers prioritize development and maintenance processes [5].

* Corresponding author.
Received: June 25th, 2025. Revised: October 10th, 2025. Accepted: December 11th, 2025.
Available online: January 15th, 2026.

DOI: https://doi.org/10.12962/j24068535.v24i1.a1333

The main challenge in classifying app reviews lies in the presence of semantic ambiguity, as a single review often expresses more than one aspect simultaneously [6]. To address this issue, multi-label text classification is increasingly used to help parse sentences and capture multiple aspects more effectively. The use of multi-label classification in this research provides several important advantages. First, this approach enables a single user review to be assigned more than one category simultaneously, reflecting the complexity and characteristics of user generated content, which often includes elements such as bug reports, feature requests, and user experiences. Second, multi-label classification helps reduce bias and misclassification errors by preventing the forced assignment of a review into a single category. Additionally, by integrating active learning, the system can identify and prioritize the most informative data for labeling, thereby significantly reducing the manual annotation burden without sacrificing model performance [7].

However, previous studies show that the presence of multiple aspects within a single review often results in overlapping meanings that are not explicitly addressed in many earlier works. Although some studies have employed transformer-based models such as BERT for review classification, most have not thoroughly evaluated or analyzed semantic ambiguity that arises when multiple categories of software requirements appear within a single review. Moreover, the use of multi-label BERT to handle ambiguity in app reviews remains limited and has not been widely examined comprehensively. Many earlier studies focus on a single dataset source, leaving variations in review contexts insufficiently captured. As a result, these studies have not fully addressed the challenge of category overlap such as between bug reports, feature requests, and user experiences, nor have they provided analyses of the causes of misclassification, which commonly occur in categories with high ambiguity. This condition highlights a research gap that requires a deeper and more systematic approach to addressing semantic ambiguity.

To provide automated support for app review analysis, this study proposes the automatic classification of reviews based on the types of information they contain. This research develops, tests, and compares various classification models to categorize reviews into four main types. Bug reports cover issues within the application that require correction, such as crashes, incorrect behavior, or performance problems. Feature requests contain user suggestions regarding new features or enhancements to existing ones. This study employs a multi-label classification approach, where a single review (instance) may contain more than one label simultaneously. Unlike Aspect Based Sentiment Analysis (ABSA), which focuses on identifying specific aspects and their sentiment polarity, the objective of this approach is to classify reviews into relevant software requirement categories without separating sentiment polarity [8]. This method is highly suitable for capturing the multifaceted nature of user feedback, which often reflects overlapping software needs within a single review. The user experiences category generally includes elements of “usability” and “feature-related feedback,” as identified by Pagano and Maalej [9]. BERT is selected as the transformer model for classification due to its capability in multi-label classification tasks and its ability to leverage transfer learning, as demonstrated in [10].

This study provides three main contributions. First, it constructs an annotated app review dataset for software requirement classification by assigning multi-label annotations to previously unlabeled reviews. Second, it applies multi-label BERT-based transfer learning to classify software requirements, particularly in cases where a single review contains more than one type of requirement. Third, it addresses the challenge of semantic ambiguity in user reviews by leveraging BERT’s strong contextual understanding capabilities.

The remainder of this paper is structured as follows. Section II presents the literature review of relevant studies. Section III explains the methodology used in this research. Section IV provides the experimental results and a comparison of the performance of the evaluated classification techniques. The findings, implications, and conclusions are discussed in Section V.

2. Literature Review

Several studies have explored the use of BERT and other transformer-based models for app review classification, showing strong potential for improving software requirement extraction. The work of Bhamare and Prabhu (2021) [6] introduced three multi-label ABSA approaches and demonstrated high accuracy on SemEval datasets.

However, these datasets are relatively clean and structured, making the methods less applicable to real-world app reviews, which contain informal expressions, mixed intentions, and semantic overlap. Similarly, the study by Maalej and Nabil (2015) [9] classified app reviews into four requirement categories using probabilistic techniques, metadata, and NLP-based features, but traditional models such as bag of words and string matching lack the contextual understanding required to capture ambiguous or multi-aspect feedback. Research by Araujo et al. (2022) [4] further demonstrated the superiority of neural language models such as BERT and RoBERTa over traditional text representation approaches; yet the study did not consider multi-label classification nor evaluate semantic ambiguity between requirement categories.

Other transformer based studies also reveal similar limitations. de Lima et al. (2025) [5] used DistilBERT for real-time issue detection, but the focus was on monitoring trends rather than classifying overlapping categories. Hadi and Fard (2023) [11], along with Roy and Biswas (2024) [12] and Azam (2023) [13], showed that PLMs outperform earlier approaches across multiple datasets and training scenarios, but these works largely emphasize performance metrics without examining misclassification patterns caused by overlapping requirement types. Interpretability focused approaches such as IARCT by Biswas et al. (2024) [14] improve transparency but do not analyze why certain categories, especially user experiences, frequently exhibit ambiguity and low classifier performance. Traditional approaches such as associative classification in Al-Hawari et al. (2020) [15] can uncover explicit rules but are limited in modeling contextual semantics and are unsuitable for multi-label settings. Meanwhile, studies like Wang et al. (2020) [16] and Azam (2023) [13] focus on semantic similarity or emotional classification, which, although useful, do not address the multi-aspect nature of software requirement extraction.

Taken together, prior studies share several key limitations. First, most works do not explicitly address semantic ambiguity, where a single review may simultaneously express multiple needs such as bug reports, feature requests, and user experiences. Second, multi-label classification using BERT for app review ambiguity remains underexplored, despite being more aligned with the actual structure of user feedback. Third, the majority of studies rely on single source datasets and do not evaluate model generalization across different application domains. Fourth, misclassification causes, error patterns, and the difficulty of overlapping categories, especially user experiences, have not been systematically analyzed. Finally, existing work focuses more on accuracy improvement rather than understanding the linguistic characteristics that contribute to category confusion.

These recurring limitations highlight a clear research gap: no previous study has comprehensively integrated multi-label BERT-based classification with explicit analysis of ambiguity and category overlap in app reviews. The absence of such analysis limits the ability of prior models to handle real-world review data, which is inherently noisy, multi-aspect, and context-dependent. Therefore, the present research advances prior work by constructing a more representative multi-label dataset sourced from two different corpora, applying multi-label BERT-based transfer learning to classify reviews containing multiple overlapping requirements, and analyzing performance specifically on ambiguous categories such as user experiences. This study not only applies transformer models but also addresses the deeper linguistic and semantic challenges that previous studies have overlooked, offering a more robust and realistic understanding of app review classification for software requirement extraction.

3 Methodology

This study was conducted through a series of systematic stages designed to support the multi-label classification process on app user review data. These stages include: data collection, manual multi-label annotation, data preprocessing, data splitting, application of the classification method, and model performance evaluation. The entire research process is visually illustrated in the methodology flow diagram, as shown in Fig. 1 below.

3.1 Data Collection dan Manual Multi-Label Annotation

This study utilizes two primary datasets, namely the Maalej Dataset obtained from the Mendeley Data platform [17] and the Kaggle dataset titled “Top 20 Play Store App Reviews (Daily Update)”. Both datasets contain mobile application user reviews and serve as the foundational corpus for the multi-label classification task. The Maalej Dataset consists of 3,891 entries, while the Kaggle dataset comprises 199,957 entries. Considering the limited size

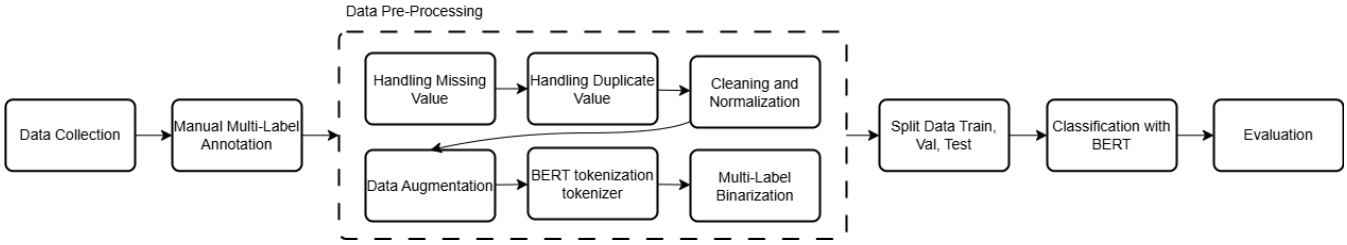

Fig. 1: Research metodology.

Table 1: Dataset description.

No	Structure and Characteristics	Data Description 1	Data Description 2
1	Dataset name	Maalej Dataset	Top 20 Play Store App Reviews (Daily Update) – all combined
2	Source	Mendeley Data	Kaggle
3	Link	https://data.mendeley.com/datasets/5fk732vkwr/2	https://www.kaggle.com/datasets/odins0n/top-20-play-store-app-reviews-daily-update
4	Description	Id_Num, Review	reviewId,content,score,app
5	Number of rows	3691	199967
6	Data type	Int dan object	object
7	Summary	Dataset contains user reviews of mobile apps annotated with categories such as feature request, bug report, praise, etc.	Dataset contains daily reviews of the top 20 apps on the Google Play Store. The data is raw and has not been annotated.

of the Maalej Dataset, both datasets were merged to construct a more representative and diverse collection of app reviews. The detailed characteristics of each dataset are presented in Table 1.

Prior to the merging process, the structures of the two datasets were standardized to match the Maalej Dataset format, which includes the attributes id and review. Additionally, four new label attributes label_1, label_2, label_3, and label_4 were introduced as placeholders for multi-label categories. Upon completing the structural harmonization, manual annotation was conducted following a predefined set of annotation guidelines.

As part of the data curation process, reviews containing fewer than 15 words were removed. This threshold was established to reduce noise, as very short reviews generally provide limited semantic information and are therefore unsuitable for meaningful multi-label assignment. Additionally, sufficient linguistic context is required to maintain annotation stability, particularly because distinguishing between categories such as bug reports and user experiences necessitates richer textual cues. The use of such a minimum-length constraint is also consistent with previous app review mining studies, especially when the classification task aims to capture deeper semantic relationships within user-generated content.

The wordcloud visualization in Fig. 2 illustrates the dominant tokens across each label category. The consistent presence of the word app across all categories confirms that user reviews predominantly revolve around their experience with the application. The Bug Reports category is characterized by problem centric terms such as problem, issue, and crash, reflecting functional failures that disrupt normal usage. Conversely, Feature Requests are dominated by intentional verbs such as want, need, and the polite marker please, indicating users' expectations for new or improved features. Ratings contain evaluative adjectives such as good, great, and love, whereas User Experiences encompass situational terms like time, show, and back. The overlap of tokens such as cant and account across all categories highlights the presence of semantic ambiguity one of the central motivations for applying a multi-label classification approach.

3.1.1 Annotation Guidelines

The manual annotation process in this study was conducted based on an explicitly defined and systematically structured set of guidelines. These guidelines were developed to ensure consistency across annotators, reduce

Fig. 2: Wordcloud for each label.

subjectivity, and minimize potential bias arising from the use of keyword-based indicators. The primary keyword list employed during annotation, presented in Table 2, was adapted from prior studies on app review classification. The annotation guidelines are described as follows:

1. Identification of Dominant Keywords

Annotators first examine the presence of keywords associated with the four main categories, bug reports, feature requests, user experiences, and ratings. Identified keywords must provide sufficiently strong semantic indications that the content of the review corresponds to a given category. This identification is conducted across the entire review text, rather than limited to the beginning or end of sentences, to ensure a comprehensive understanding of the review context.

2. Assignment of Multiple Labels When More Than One Category Is Indicated

If a review contains keywords from multiple categories, all relevant categories are assigned as labels. This approach acknowledges that app reviews often address more than one issue simultaneously, such as a bug complaint combined with a feature request or comments on overall user experience. As a result, the model is trained to recognize complex interactions between different aspects within user generated content.

3. Contextual Interpretation to Avoid Over-Tagging

Although the annotation process relies on keywords, label assignment is not performed mechanically. Annotators must evaluate the contextual meaning surrounding each keyword. If a keyword appears in a context unrelated to the intended category, the corresponding label is not assigned. This contextual interpretation prevents misclassification caused by linguistic ambiguity and ensures that labels are assigned only when the surrounding context supports the category.

4. Handling Reviews Without Keywords or Sufficient Semantic Indicators

If a review does not contain relevant keywords and lacks strong semantic cues that can be mapped to any category, the review is excluded from the annotated dataset. This decision is based on the understanding that reviews with insufficient contextual information tend to produce unstable labels and may negatively affect model learning

Table 2: Keyword manual multi-label annotation.

Review Type	Keyword
Bug reports	bug, fix, problem, issue, defect, crash, solve
Feature requests	add, please, could, would, hope, improve, miss, need, prefer, request, should, suggest, want, wish
User Experiences	help, support, assist, when, situation
Ratings	great, good, nice, very, cool, love, hate, bad, worst

Table 3: Dataset after manual multi-label annotation.

No	Structure and Characteristics	Maalej Dataset	Kaggle Dataset	Combine Dataset
1	Attributes	id review Label_1 Label_2 Label_3 Label_4	id review Label_1 Label_2 Label_3 Label_4	id review Label_1 Label_2 Label_3 Label_4
2	Number of Data After Annotation	1394	30159	31553
3	Number of Data per Label	Bug Feature : 419 Feature Request : 708 User Experiences : 594 Ratings : 952	Bug Feature : 11592 Feature Request : 14748 User Experiences : 15050 Ratings : 13161	Bug Feature : 12011 Feature Request : 15456 User Experiences : 15644 Ratings : 14113

quality, particularly in a multi-label classification setting that requires clear semantic relationships between text and labels.

After the annotation process was completed, the Maalej Dataset and Kaggle Dataset, which contained mobile app reviews labeled according to relevant categories (such as bug reports, feature requests, user experiences, and ratings), were obtained. These datasets were then merged into a Combined Dataset and prepared for use in multi-label classification model experiments. A more detailed explanation of the structure and characteristics of the annotated datasets can be found in Table 3, which provides information on the processed data, including the number of reviews in each category and the data types used. This dataset is the result of a meticulous annotation process, where each mobile app review was labeled with one or more categories based on the keywords found within the review. With this dataset, the multi-label classification model is ready to be tested, enabling the model to analyze and classify app reviews more accurately and efficiently.

3.2 Data Preprocessing

The data preprocessing stage was carried out to remove irrelevant information and transform the app reviews into a structured linguistic representation [18]. This stage serves as an initial step in addressing ambiguity in software requirement classification based on user reviews using a multi-label BERT transfer learning approach. In this study, the data preprocessing process was systematically conducted through the following steps:

1. Cleaning and Normalization Data

The initial step in preprocessing involves identifying and handling missing values and duplicate entries in the dataset. Once the review data has been cleaned of empty and duplicate values, the next stage is text cleaning and normalization to ensure format consistency and readability by the classification model. The following are the cleaning and normalization processes applied in this study:

- Removal of Non-Alphabetic Characters and Punctuation: Non-alphabetic characters, numbers, and irrelevant symbols were removed to reduce noise.
- Lowercasing: All text was converted to lowercase to maintain consistency [19].
- Stopword Removal: Common words that contributed minimally to the contextual understanding were selectively removed using the NLTK stopword repository [20].

Table 4: Augmented data.

No	Clean Review	Augmented Review
1	besides occasional crash amazing product tons potential depending work	besides episodic crash amazing product tons potential depending work
2	could great app predictable full bugs unpredictable able check take screen shot boarding pass print backup copy may able access need	could great app predictable full bugs unpredictable capable check take screen shot boarding pass print backup copy may capable access need
3	use love app working new update pages wont scroll downnone different tabs workits frozen please fix asap	use lovemaking app working new update pages wont scroll downnone different tabs workits frozen please fix asap

As part of the preprocessing strategy, reviews containing fewer than 15 words were excluded to improve annotation reliability and ensure that each text sample provided sufficient semantic information for multi-label classification. Extremely short reviews tend to express generic sentiments such as “great app” or “bad update” which offer limited interpretive value and rarely contain cues necessary to distinguish between categories such as bug reports, feature requests, or user experiences. Moreover, insufficient linguistic context often leads to unstable annotations, thereby reducing the consistency required for multi-label learning. The application of this threshold aligns with established practices in app review mining, particularly when the objective is to extract fine-grained requirement information.

Additionally, although BERT-large is capable of processing sequences up to 512 tokens, this study employed a maximum sequence length of 128 tokens. Statistical analysis of the dataset indicated that the vast majority of reviews contained fewer than 100 tokens, making 128 tokens sufficient to retain nearly all meaningful content without significant truncation. This configuration also enhances computational efficiency, as BERT-large is resource-intensive; shorter input sequences reduce training time and memory usage while preserving contextual richness. Furthermore, limiting sequence length minimizes excessive padding, which can introduce noise and increase the risk of overfitting. Thus, setting the sequence length to 128 provides an optimal balance between efficiency, semantic coverage, and model generalization.

By integrating these preprocessing considerations thresholding for semantic adequacy, careful stopword handling, and optimized sequence length the dataset is prepared to support a robust and reliable multi-label classification process in subsequent stages of the study.

2. Data Augmentation

The next stage involved data augmentation, which aims to generate synthetic data from the original dataset through transformations that preserve the semantic meaning and associated labels [21]. In this study, two augmentation techniques were applied: synonym replacement and word shuffling. Table 4 is an example of data that has undergone augmentation, as described in the previous section:

3. BERT Tokenization

In the preprocessing stage, tokenization was performed using the BERT tokenizer, which converts each token in the app review into an embedding represented as a numerical vector. These vectors capture the semantic meaning of the tokens, such that tokens with similar meanings have closely aligned vector representations [22]. This transformation enables the model to more accurately interpret the context and meaning of words in numerical form, thus enhancing its understanding of textual data. Some examples of the application of the BERT tokenizer are shown in Table 5 :

4. Multi-Label Binarization

In this study, the classification approach used is multi-label classification, where each app review sample can be associated with more than one label category. Therefore, a transformation process is required to convert the original labels into an appropriate format, specifically in the form of a binary vector. This process is known as multi-label binarization, which aims to convert each set of labels into a fixed-dimensional binary vector. Each element in the

Table 5: BERT tokenizer data.

No	Augmented Review	Bert Tokenizer
1	besides episodic crash amazing product tons potential depending work	[101, 4661, 4958, 19565, 14808, 5823, 6429, 4031, 6197, 4022, 5834, 2147, 102, 0]
2	could great app predictable full bugs unpredictable capable check take screen shot boarding pass print backup copy may capable access need	[101, 2071, 2307, 10439, 21425, 2440, 12883, 21446, 5214, 4638, 2202, 3898, 2915, 9405, 3413, 6140, 10200, 6100, 2089, 5214, 3229, 2342, 102, 0]
3	use lovemaking app working new update pages wont scroll downnone different tabs workits frozen please fix asap	[101, 2224, 2293, 12614, 10439, 2551, 2047, 10651, 5530, 2180, 2102, 17186, 2091, 8540, 2063, 2367, 21628, 2015, 2147, 12762, 7708, 3531, 8081, 17306, 2361, 102, 0]

Table 6: Raw data.

ID	Review	Label_1	Label_2	Label_3	Label_4
1	Besides the occasional crash, this is an amazing product with tons of potential depending on how you work with it!	Bug reports	User Experiences	Ratings	
2	This could be a great app if it was predictable, but it is full of bugs and is unpredictable. if you are able to check in, take a screen shot of your boarding pass or print a backup copy, because you may not be able to access it when you need it most.	Bug reports	Feature requests	User Experiences	Ratings
3	Use to love this app but it's not working after new update. Pages won't scroll up or down…none of the different tabs work…it's frozen! Please fix ASAP!!!	Bug reports	Feature requests	Ratings	

vector corresponds to a specific class category, such as feature requests, bug reports, ratings, and user experiences. A value of 1 in a given position of the vector indicates the presence of that label in the review, while a value of 0 indicates its absence. Table 6 shows the raw data or the data that has not undergone preprocessing :

3.3 Model Training and Evaluation

After completing the preprocessing stage, the dataset was used to train a BERT-based multi-label classification model. The overall training process consists of three main stages: data splitting, model training, and evaluation, as illustrated in Fig. Fig. 1.

1. Split Data: Train, Validation, and Test

The cleaned dataset was divided into three subsets: 70% for training, 15% for validation, and 15% for testing. Details of the final data distribution are presented in Table 7. Prior to splitting, duplicate and empty entries were removed, resulting in adjusted dataset sizes. This approach enhances the model's ability to generalize and reduces potential bias during training and evaluation.

2. Classification with BERT

BERT (Bidirectional Encoder Representations from Transformers) is a Transformer-based language model introduced by Devlin et al. (2018) [23]. As illustrated in Fig. 3, the model is constructed using stacked encoder blocks, each consisting of a multi-head self-attention mechanism and a feedforward neural network [24]. The self-

Table 7: Split dataset.

No	Data	Split Data	Number of Data
1	Training Data	70%	22087
2	Validation Data	15%	4733
3	Test Data	15%	4733

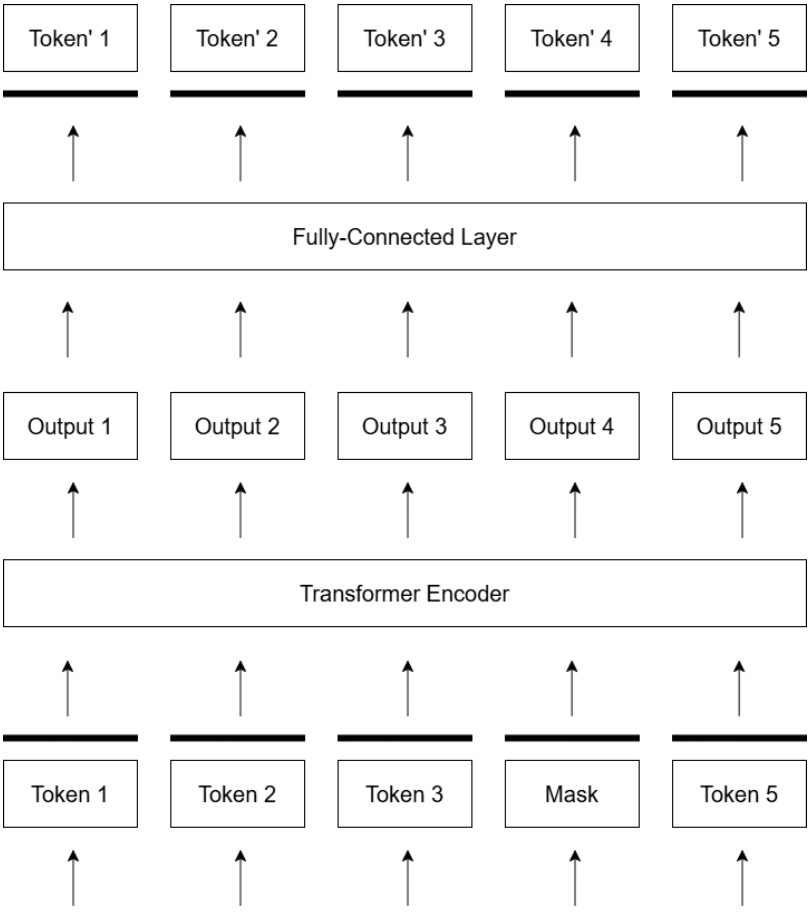

Fig. 3: BERT base architecture.

attention mechanism allows each token to attend to every other token, enabling the model to capture complex semantic relationships within a sentence.

During pretraining, BERT is optimized using two self-supervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM enables the model to understand bidirectional context, while NSP strengthens inter-sentence coherence modeling [23]. Once pretrained, BERT can be fine-tuned for downstream tasks such as the multi-label classification used in this study.

Although the illustration above shows a generic Transformer encoder structure, this study specifically employs BERT-large-uncased, which contains 24 encoder layers, 16 attention heads, and 340 million parameters, providing higher representational capacity and greater contextual depth for semantic understanding.

To adapt BERT for multi-label classification, modifications were applied to the classification head. Because the task allows multiple labels to co-occur in a single review, the output layer uses sigmoid activation, not softmax. Softmax assumes mutually exclusive classes, whereas sigmoid allows independent probability estimation for each label (Bug Reports, Feature Requests, User Experiences, Ratings).

During optimization, the model computes errors using Binary Cross-Entropy with Logits Loss (BCEWith-LogitsLoss), the standard loss function for multi-label neural models. This loss integrates sigmoid and binary cross-entropy into a stable computation process and is widely applied in multi-label Transformer architectures.

Table 8: Hyperparameter settings for fine-tuning BERT-large.

Hyperparameter	Value
“Model Architecture”	“BERT-large-uncased”
“Batch Size”	“32”
“Learning Rate”	“2e-5”
“Max Sequence Length”	“128 tokens”
“Epochs”	“3”
“Warmup Steps”	“100”
“Weight Decay”	“0.01”
“Optimizer”	“AdamW”
“Scheduler”	“Linear schedule with warmup”
“Loss Function”	“BCEWithLogitsLoss (multi-label)”
“Activation Function”	“Sigmoid”

To ensure reproducibility and address the reviewer’s concern regarding missing configuration details, all hyperparameters used for fine-tuning BERT-large are summarized Table 8 below:

This configuration allows the model to learn expressive features effectively while maintaining computational efficiency. The use of BCEWithLogitsLoss and sigmoid activation ensures accurate probability estimation across multiple co-occurring requirement categories, addressing the inherent ambiguity found in software application reviews.

3. Evaluation

The evaluation of the multi-label classification model in this study was conducted using the metrics Precision, Recall, and F1-score to assess the effectiveness of the model in classifying software requirements based on app reviews. Precision measures the model’s accuracy in predicting the correct labels, specifically the proportion of predicted positive labels that are truly relevant [18]. Recall evaluates the model’s ability to identify all actual positive labels, which is particularly important to ensure that no critical software requirement is overlooked [18]. Since multi-label classification involves predicting multiple labels simultaneously, the F1 score defined as the harmonic mean of Precision and Recall provides a balanced view of the model’s performance in terms of both accuracy and completeness [18]. By applying these metrics, the study effectively assesses how well the implemented BERT transfer learning model addresses textual ambiguity in app reviews and delivers accurate and comprehensive software requirement classifications. Below are the formulas used for the evaluation metrics:

Precision measures the proportion of correctly predicted positive labels out of all labels predicted as positive by the model.

$$Precision_{micro} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i + FP_i)} \tag{1}$$

Recall measures the proportion of correctly predicted positive labels out of all actual positive labels.

$$Recall_{micro} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i + FN_i)} \tag{2}$$

The **F1-score** is the harmonic mean of Precision and Recall. It balances both metrics, especially when there is a trade-off between them.

$$F1_{micro} = 2 \times \frac{Precision_{micro} \times Recall_{micro}}{Precision_{micro} + Recall_{micro}} \tag{3}$$

Table 9: Performance evaluation , Maalej Dataset.

Label	Precision	Recall	F1-Score
Bug Reports	0.67	0.87	0.76
Feature Requests	0.80	0.86	0.83
Ratings	0.86	0.79	0.82
User Experiences	0.48	0.82	0.60
Micro Average	0.69	0.83	0.75
Macro Average	0.70	0.83	0.75
Weighted Average	0.73	0.83	0.77
Samples Average	0.74	0.85	0.75

Table 10: Performance evaluation , Kaggle Dataset.

Label	Precision	Recall	F1-Score
Bug Reports	0.95	0.96	0.95
Feature Requests	0.99	0.91	0.95
Ratings	0.98	0.96	0.97
User Experiences	0.89	0.73	0.80
Micro Average	0.95	0.88	0.92
Macro Average	0.95	0.89	0.92
Weighted Average	0.95	0.88	0.91
Samples Average	0.95	0.90	0.91

Table 11: Performance evaluation , Combined Dataset.

Label	Precision	Recall	F1-Score
Bug Reports	0.94	0.97	0.96
Feature Requests	0.99	0.93	0.96
Ratings	0.98	0.97	0.97
User Experiences	0.89	0.72	0.80
Micro Average	0.95	0.89	0.92
Macro Average	0.95	0.90	0.92
Weighted Average	0.95	0.89	0.92
Samples Average	0.95	0.90	0.91

4. Result and Discussion

In this study, we evaluated the performance of a BERT-based multi-label classification model using three dataset configurations: the Maalej dataset, the Kaggle dataset, and the combined dataset. These experiments provide comprehensive insights into how dataset size, linguistic variation, semantic complexity, and category ambiguity influence the performance of transformer-based models in software-requirement classification.

The evaluation results across the three dataset configurations Maalej, Kaggle, and the combined dataset, provide deep insights into how dataset size, variability, semantic richness, and category ambiguity affect multi-label classification performance. The model shows strong performance in structured and explicit categories such as Bug Reports and Ratings, but continues to face significant challenges in handling semantically ambiguous categories such as User Experiences.

The performance of the model on the Maalej dataset (Table 9) highlights the constraints caused by limited training data. The Bug Reports category exhibits high recall (0.87) but low precision (0.67), suggesting that the model frequently over predicts this label. This indicates that the small dataset does not provide enough negative

examples for the model to develop fine grained discriminative capability. Meanwhile, Feature Requests (F1 = 0.83) and Ratings (F1 = 0.82) perform moderately well due to clearer lexical cues commonly used in these categories, such as “add”, “wish”, or “rate.”

However, the User Experiences category shows the weakest performance (F1 = 0.60), driven by low precision (0.48). This suggests substantial misclassification, where reviews without experiential content are mislabeled as User Experiences. This behaviour is consistent with the broad semantic boundaries of the category, which includes subjective impressions, usability comments, and general satisfaction. Aggressive preprocessing (e.g., removing short reviews) likely also removed concise yet meaningful experiential expressions, worsening category imbalance.

A major improvement is observed in the Kaggle dataset (Table 10), which contains more than 21,000 training samples. The model benefits from richer linguistic patterns and contextual variety, yielding near-perfect performance for Bug Reports (F1 = 0.95), Feature Requests (F1 = 0.95), and Ratings (F1 = 0.97). Precision increases substantially across all categories, confirming that the model becomes more confident and accurate in identifying explicit requirement expressions. Even User Experiences improves to F1 = 0.80, although it remains the lowest-performing category. This persistent gap reflects the intrinsic semantic overlap of this category with others, e.g., experiential comments frequently overlap with implicit ratings, usability suggestions, or bug-like issues.

The combined dataset (Table 11) yields the most stable and consistent performance across categories. The fusion of domain-specific Maalej data and lexically diverse Kaggle data enables the model to learn robust contextual representations. The Bug Reports, Feature Requests, and Ratings categories show highly reliable performance (F1 = 0.96–0.97). The User Experiences category also improves relative to the Maalej dataset alone but remains at F1 = 0.80. This indicates that despite increased data diversity, the category’s semantic ambiguity fundamentally limits classification performance.

Across all datasets, several persistent error patterns are observed. First, implicit meaning frequently causes errors, as experiential feedback may not contain explicit keywords. Second, label co occurrence is common; many reviews simultaneously express bugs, suggestions, and emotional responses. Third, class imbalance particularly in the Maalej dataset introduces bias where the model tends to over predict high frequency categories. Fourth, subjective expressions in User Experiences often resemble mild complaints or praise, confusing the classifier.

The impact of data augmentation was more evident in the Maalej dataset. Synonym replacement improved recall for minority classes by introducing lexical variety, whereas word shuffling augmentation proved less effective due to syntactic distortion. More advanced augmentation (e.g., contextualized BERT masking) may preserve semantics better and provide more beneficial improvements.

Macro, micro, weighted, and sample level averages also show important patterns. Micro average scores reflect strong overall performance across labels, but macro average results reveal large variance due to the consistently lower User Experiences performance. Weighted averages highlight how dominant categories influence the overall result, emphasizing the importance of per label analysis rather than relying on aggregated metrics alone.

Overall, these findings illustrate that dataset size, linguistic richness, and semantic clarity are critical determinants of model performance. The model is highly effective for structured and explicit requirement categories but remains limited in capturing nuanced experiential feedback. This aligns with prior research emphasizing the inherent ambiguity of user generated content. Practical implications include the model’s strong suitability for automated extraction of bug reports and feature requests, while subjective categories may require complementary techniques such as hierarchical classification, sentiment intent disentanglement, or transformer models with label dependency modeling.

Future research should explore semi supervised learning to expand labeled data for ambiguous categories, hierarchical multi-label architectures to capture label relationships, and improved annotation guidelines to ensure higher inter annotator agreement, particularly for categories with broad semantic scope such as User Experiences.

5. Conclusion

This study demonstrates that the multi-label classification approach using BERT-based transfer learning is effective in addressing ambiguity in app reviews for software requirement extraction. The integration of the Maalej and Kaggle datasets, combined with manual annotation, data augmentation, and structured preprocessing, enables the model to classify reviews into four categories bug reports, feature requests, user experiences, and ratings with strong overall performance, reflected in high micro-average F1-scores. The model performs particularly well in identifying bug reports, feature requests, and ratings, although classification of user experiences remains challenging due to the highly varied and implicit nature of experiential expressions as well as the semantic overlap with other categories. Despite these promising findings, several limitations remain. The threshold of 15 words used during preprocessing, while improving annotation stability, may exclude potentially meaningful but concise reviews. Dependence on keyword-based annotation introduces bias and may fail to capture nuanced or implicit meanings in user comments. Additionally, the study employs only a single BERT-large model without exploring ensemble techniques or alternative architectures that could improve robustness, and the flat label structure does not account for potential hierarchical relationships between categories. Given these limitations, future research should explore active learning to enhance annotation quality, ensemble or hybrid Transformer models to increase predictive stability, and hierarchical multi-label methods to better represent structured relationships among review categories. Semi-supervised and weakly supervised approaches also present opportunities to leverage large volumes of unlabeled review data, while more adaptive preprocessing techniques are needed to balance noise reduction with semantic preservation. Overall, this research highlights both the effectiveness and the areas of improvement for BERT-based multi-label classification in extracting software requirements from ambiguous user reviews and sets a strong foundation for more advanced methodologies in future studies.

CRediT authorship contribution statement

S. T. Hallatu: Methodology, Software, Validation, Writing – Original Draft. **M. J. Gorter:** Validation, Formal analysis, Investigation, Writing – Original Draft. **A. B. Jati:** Resources, Data Curation, Writing – Original Draft. **D. Purwitasari:** Writing – Review & Editing. **C. Fatichah:** Writing – Review & Editing. **H. Tsaniya:** Writing – Review & Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This study received partial funding from Institut Teknologi Sepuluh Nopember (ITS) through the 2025 Institutional Development Research Grant and Departmental Research Grant programs.

Data availability

The dataset was openly provided [https://data.mendeley.com/datasets/5fk732vkwr/2], [https://www.kaggle.com/datasets/odins0n/top-20-play-store-app-reviews-daily-update].

Declaration of Generative AI and AI-assisted Technologies in The Writing Process

The authors used generative AI to improve the writing clarity of this paper. They reviewed and edited the AI-assisted content and take full responsibility for the final publication.

References

[1] X. Li, B. Zhang, Z. Zhang, and K. Stefanidis, “A sentiment-statistical approach for identifying problematic mobile app updates based on user reviews,” *Information (Switzerland)*, vol. 11, no. 3, 2020, doi: 10.3390/info11030152.

[2] B. Ma, Y. D. Wong, C. C. Teo, and Z. Wang, “Enhance understandings of Online Food Delivery's service quality with online reviews,” *Journal of Retailing and Consumer Services*, vol. 76, no. September2023, p. 103588, 2024, doi: 10.1016/j.jretconser.2023.103588.

[3] G. S. Sunarko, Wasino, and T. Sutrisno, “Klasterisasi Sentimen Ulasan Pengguna Aplikasi Bca Mobile Pada Platform Google Play Store Dengan Algoritma K-Means Clustering,” *Jurnal Ilmu Komputer dan Sistem Informasi*, vol. 11, no. 1, 2023, doi: 10.24912/jiksi.v11i1.24145.

[4] A. F. Araujo, M. P. Gôlo, and R. M. Marcacini, "Opinion mining for app reviews: an analysis of textual representation and predictive models," *Automated Software Engineering*, vol. 29, no. 1, May 2022, doi: 10.1007/s10515-021-00301-1.

[5] V. M. A. de Lima, J. R. Barbosa, and R. M. Marcacini, "Issue detection and prioritization based on mobile application reviews," *Software Quality Journal*, vol. 33, no. 1, p. 6, Mar. 2025, doi: 10.1007/s11219-024-09703-2.

[6] B. R. Bhamare and J. Prabhu, "A multilabel classifier for text classification and enhanced BERT system," *Revue d'Intelligence Artificielle*, vol. 35, no. 2, pp. 167–176, 2021, doi: 10.18280/ria.350209.

[7] M. B. Messaoud, I. Jenhani, N. B. Jemaa, and M. W. Mkaouer, "A Multi-label Active Learning Approach for Mobile App User Review Classification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 805–816, 2019, doi: 10.1007/978-3-030-29551-6_71.

[8] X. Chen, Y. Yin, and T. Feng, "Multi-Label Text Classification Based on BERT and Label Attention Mechanism," *Proceedings - 2023 Asia-Pacific Conference on Image Processing, Electronics and Computers, IPEC 2023*, pp. 386–390, 2023, doi: 10.1109/IPEC57296.2023.00073.

[9] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," *2015 IEEE 23rd International Requirements Engineering Conference, RE 2015 - Proceedings*, pp. 116–125, 2015, doi: 10.1109/RE.2015.7320414.

[10] D. Kici, G. Malik, M. Cevik, D. Parikh, and A. Başar, "A BERT-based transfer learning approach to text classification on software requirements specifications," *Proceedings of the Canadian Conference on Artificial Intelligence*, pp. 1–13, 2021, doi: 10.21428/594757db.a4880a62.

[11] M. A. Hadi and F. H. Fard, *Evaluating pre-trained models for user feedback analysis in software engineering: a study on classification of app-reviews*, vol. 28, no. 4. 2023. doi: 10.1007/s10664-023-10314-x.

[12] G. T. Roy and D. Biswas, "Exploring Transformer and Recurrent Neural Models for Sentiment Analysis of Mobile App Reviews," *2nd International Conference on Information and Communication Technology, ICICT 2024*, pp. 209–213, 2024, doi: 10.1109/ICICT64387.2024.10839678.

[13] F. Azam, M. w. L. Gloire, N. Priyadarshi, and S. Kumari, "Text Classification into Emotional States Using Deep Learning based BERT Technique," *2023 4th IEEE Global Conference for Advancement in Technology, GCAT 2023*, pp. 1–5, 2023, doi: 10.1109/GCAT59970.2023.10353414.

[14] M. Biswas, P. R. Anish, and S. Ghaisas, "Interpretable App Review Classification with Transformers," *Proceedings - 32nd IEEE International Requirements Engineering Conference Workshops, REW 2024*, pp. 26–34, 2024, doi: 10.1109/REW61692.2024.00009.

[15] A. Al-Hawari, H. Najadat, and R. Shatnawi, "Classification of application reviews into software maintenance tasks using data mining techniques," *Software Quality Journal*, vol. 29, no. 3, pp. 667–703, 2021, doi: 10.1007/s11219-020-09529-8.

[16] X. Wang, W. Zhang, S. Lai, C. Ye, and H. Zhou, "The Use of Pretrained Model for Matching App Reviews and Bug Reports," *IEEE International Conference on Software Quality, Reliability and Security, QRS*, pp. 242–251, 2022, doi: 10.1109/QRS57517.2022.00034.

[17] A. Al-Hawari, *A dataset of Mobile application reviews for classifying reviews into software Engineering's maintenance tasks using data mining techniques*. (2019). Mendeley Data. doi: 10.17632/5fk732vkwr.2.

[18] M. Arief and M. B. M. Deris, "Text Preprocessing Impact for Sentiment Classification in Product Review," in *2021 6th International Conference on Informatics and Computing, ICIC 2021*, Institute of Electrical, Electronics Engineers Inc., 2021. doi: 10.1109/ICIC54025.2021.9632884.

[19] Y. Zhou, X. Wang, and K. F. Yuen, "Sustainability disclosure for container shipping: A text-mining approach," *Transport Policy*, vol. 110, pp. 465–477, Sept. 2021, doi: 10.1016/j.tranpol.2021.06.020.

[20] G. Liu, M. Boyd, M. Yu, S. Z. Halim, and N. Quddus, "Identifying causality and contributory factors of pipeline incidents by employing natural language processing and text mining techniques," *Process Safety and Environmental Protection*, vol. 152, pp. 37–46, Aug. 2021, doi: 10.1016/j.psep.2021.05.036.

[21] C. Shorten, T. M. Khoshgoftaar, and B. Furht, "Text Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00492-0.

[22] A. K. Nandanwar and J. Choudhary, "Contextual Embeddings-Based Web Page Categorization Using the Fine-Tune BERT Model," *Symmetry*, vol. 15, no. 2, Feb. 2023, doi: 10.3390/sym15020395.

[23] Z. Wang, L. Wang, C. Huang, S. Sun, and X. Luo, "BERT-based chinese text classification for emergency management with a novel loss function," *Applied Intelligence*, vol. 53, no. 9, pp. 10417–10428, May 2023, doi: 10.1007/s10489-022-03946-x.

[24] B. Yu, C. Deng, and L. Bu, "Policy Text Classification Algorithm Based on Bert," in *Proceedings - 2022 11th International Conference of Information and Communication Technology, ICTech 2022*, Institute of Electrical, Electronics Engineers Inc., 2022, pp. 488–491. doi: 10.1109/ICTech55460.2022.00103.