Neural Computing and Applications (2023) 35:22487–22509
https://doi.org/10.1007/s00521-023-08833-1

ORIGINAL ARTICLE

MNoR-BERT: multi-label classification of non-functional requirements using BERT

Kamaljit Kaur[1] · Parminder Kaur[1]

Received: 6 September 2022 / Accepted: 28 June 2023 / Published online: 12 August 2023
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2023

Abstract

In the era of Internet access, software is easily available on digital distribution platforms such as app stores. The distribution of software on these platforms makes user feedback more accessible and can be used from requirements engineering to software maintenance context. However, such user reviews might contain technical information about the app that can be valuable for developers and software companies. Due to pervasive use of mobile apps, a large amount of data is created by users on daily basis. Manual identification and classification of such reviews are time-consuming and laborious tasks. Hence, automating this process is essential for assisting developers in managing these reviews efficiently. Prior studies have focused on classification of these reviews into bug reports, user experience, and feature requests. Nevertheless to date, a very few research papers have extracted Non-Functional Requirements (NFRs) present in these reviews. NFRs are considered as the set of quality attributes such as reliability, performance, security and usability of the software. Previous studies have utilized machine learning techniques to classify these reviews into their respective classes. However, it was observed that existing studies treat review classification problems as single-label classification problem, and also underestimate the contextual relationship between the words of review statements. To alleviate this limitation, the proposed research work used a transfer learning model to classify multi-label app reviews into four NFRs: Dependability, Performance, Supportability, and Usability. The proposed approach evaluates the performance of the pre-trained language model for multi-label review classification. In this paper, a set of experiments are conducted to compare the performance of the proposed model against the baseline machine learning with binary relevance and keyword based approach. We evaluated our approach over a dataset of 6000 user reviews of 24 iOS apps. Experimental results show that the proposed model outperforms state-of-the-art baseline techniques with respect to precision, recall, and F1-measure.

Keywords Requirements engineering · Non-functional requirements · Transfer learning · BERT

1 Introduction

With the rapid evolution of mobile technology, web-based software applications such as, app stores, user forums, mailing lists, wikis, newspapers, and blogs, have become more accessible. According to Statista,[1] app stores such as Google Play and App Store are considered popular platforms among internet users [1, 2]. These platforms provide a large volume of user feedback that represents valuable information, which can be analyzed to support decision making in software engineering [27]. The proliferation of available user feedback is extensively offers the opportunity to address information in novel way that need at decision-making in different software tasks, ranging from requirements engineering to maintenance and release planning. Most of the existing studies focused on the functional aspects during requirements elicitation from online reviews. Researchers have extracted informative reviews and classified them as bug reports, feature requests, and ratings [3, 10, 30, 31, 38]. Studies have indicated that only few research papers have mined Non-functional

✉ Kamaljit Kaur
Kamaljitcs.rsh@gndu.ac.in

Parminder Kaur
Parminder.dcse@gndu.ac.in

[1] Department of Computer Science, Guru Nanak Dev University, Amritsar, India

[1] https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

requirements (NFRs) from online user feedback [23, 25, 29]. Non-Functional requirements are a set of quality attributes that the system should exhibit [6]. These attributes play a vital role in software success(Bohem, 1996). Addressing these attributes are essential for achieving user satisfaction and maintaining market viability [12, 15].

Along with this trend, Crowd-based Requirements Engineering (CrowdRE) paradigm has already entered the world of requirements engineering. CrowdRE is considered feedback-based requirements engineering, where users can directly contribute to the understanding of requirements during the development of the next version of software or app release [37]. However, online user feedback has been created on an unimaginable scale. Requirements Engineers need a classification model to reduce the risk in decision-making. Although manual extraction and classification is more fine-grained and meaningful process, but manual process is time-intensive and infeasible. Therefore, automating analysis of large volume of user feedback is considered a distinguish feature of CrowdRE. This research work has analyzed user feedback from prominent crowd-sourced platforms such as Apple App Store and Google Play Store.

To address the above-mentioned drawback of manual classification, automatic classification techniques have been proposed and used in several research works (Kurtanovic and Maalej, 2017; [25, 44]. These studies reported that machine learning techniques such as, Naïve Bayes (NB), Support Vector Machine (SVM), and *K*-Nearest Neighbor (K-NN), achieve reasonable performance. Existing literature reported that DNNs (CNN, RNN) outplay machine learning techniques (Navarro-Almanza et al., 2018,[4, 22, 47]. Among the vast DNN types, CNN and RNN gained more popularity because CNN perform better in learning local patterns and RNN in sequential modeling.

However, existing models have two main drawbacks (i) these approaches do not use any language models (ii) they suffer from misclassification errors (false positive and false negative). The reason behind the high rate of misclassified instances is that keyword-based approaches view problems as bag-of-word and do not capture the contextual relationship between the words. Moreover, these models usually consider multi-label classification problems as multi-class and single-label problem. Single-label classification approaches assume that user reviews can only have single-label at a time. Users generally discuss more than one requirement in a single review, such as updating a feature increases performance. Therefore, the single-label problem becomes inadequate to classify the reviews, so we have to treat it as a multi-label classification problem. Our proposed work utilizes language model for multi-label classification problem, where reviews are associated with more than one class label.

To alleviate the aforementioned issues, this research work investigates the role of transfer learning in requirements classification. Transformers are used with their attention mechanism to eliminate the need to pre-process data. Moreover, in recent years, transfer learning has great impact on computer vision, but now Natural Language (NLP) community successfully applied transfer learning to text [18]. Natural Language models such as BERT, and XLNet reportedly produced state-of-the-art performance [17],Yang et al., 2019).

By following the same line of research, this proposed work adopts a neural language-based Bidirectional Encoder Representation from Transformers (BERT) model. BERT consists of two stages: (i) BERT model can be pre-trained on a large amount of data, with an unsupervised objective of masked language modeling and next-sentence prediction. Then, this pre-trained network is fine-tuned on task-specific labeled data. The proposed research work presents an approach MNoR-BERT (Multi-Label Non-Functional Requirements classification using BERT), that classifies multi-label user reviews into NFRs by capturing the relationships between words. The dataset used in this paper was created by Jha and Mahmoud [25]. The authors sampled 6000 user reviews from 24 iOS apps. The contributions of this work are as follows:

- Building a language-based architecture for multi-label non-functional requirements classification.
- Evaluating our model on 6000 user reviews
- The proposed model received raw inputs without pre-processing techniques.
- This is the first study that explores the applicability of BERT for automatic classification of multi-label user reviews into NFRs
- We conducted an extensive comparison between BERT with the machine and deep learning models.

The paper is organized as follows: in Sect. 2, we present the related work. In Sect. 3, we present the research methodology that has been followed. Then, proposed model and multi-label classification problem is discussed in Sects. 4 and 5, respectively. In Sect. 6, we provide the dataset, cross-validation, experimental setup, baseline models and evaluation measures. Results and discussion

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

are explained in Sect. 7. Finally, we conclude the paper in Sect. 8.

2 Related work

Requirements classification is considered as the fundamental step of RE, in which predefined categories are assigned to each requirements. The classification of app reviews into specified software requirements can help developers and app vendors deal with critical reviews for software evolution. Existing studies have discussed various approaches for user review analysis. Numerous research papers on review classification, filtering, summarization, and prioritization have been published [1, 30, 44]. However, existing studies have also focused on semantics analysis to design an effective classification algorithm, and to find which part of the review text provides better classification results. To the best of our knowledge, only a few researchers have analyzed existing reviews from the perspective of NFRs [23, 25, 29]. In this section, we discuss important and similar related works to our analysis.

Chen et al. [10] proposed the AR-Miner tool, an analytical approach for mining informative reviews, to help app developers. The tool uses a filtering technique, i.e., Expectation Maximization with Naïve Bayes to filter out "noisy and irrelevant" reviews. The authors then used topic modeling to analyze and prioritize informative reviews. Their proposed approach was evaluated on a manually labeled dataset of app reviews, and they reported the highest accuracy in terms of precision, recall and ranking quality. Guzman and Maalej [24] proposed an automated approach to help app developers to systematically analyze user opinions about single features and filter irrelevant features. The authors performed collocation and sentiment analyses to extract the fine-grained requirements present in the review. Then, the topic modeling technique is used to group the fine-grained features into more meaningful high-level features. The authors evaluated their approach on 32,210 reviews from App Store and Google Play. The reported results showed that their proposed approach managed to successfully capture and grouped the most relevant features in the review. Panichella et al. [38] proposed a semi-automated approach that classifies reviews into four types such as information seeking, information giving, and feature request and problem discovery. Maalej and Nabil [31] designed an approach based on probabilistic techniques to automatically classify user feedback into four basic types: Bug report, feature request, user experience and ratings. The authors applied Naïve Bayes, Decision tree, and Maximum Entropy to compare Binary and multi-class classifiers. The authors also concluded that combining text classification, natural language processing and pre-processing techniques significantly improves classification accuracy. McIlroy et al. [33] developed an automated approach, which analyzes the multi-labeled nature of app reviews in Google Play and App Store. A qualitative analysis revealed that 30% of the reviews contain more than one technical issue such as feature request and bug report. Groen et al. [23] have identifies statement about product quality in online reviews. The online tagging was performed to determine the quality requirements such as "usability" and "reliability." Ciurumelea et al. [13] employed machine learning techniques with Information Retrieval for the automatic categorization of user reviews in predefined classes. This automated approach analyzes and identifying these user reviews that containing relevant issues. Jha and Mahmoud (2019) proposed a dictionary-based approach which detects NFRs in user reviews. The authors evaluated their approach over dataset of 1100 reviews sampled from a set of iOS and Android applications. Their approach achieved an average precision of 70% and an average recall of 86%. Moreover, the authors tested their approach on a cut-off length of 12 words of review and reported improvements in results**.** The authors applied two machine learning algorithms NB and SVM. SVM performs better in multi-label classification than NB.

In addition, the aforementioned studies discussed traditional text classification approaches that mainly represent data with Bag-of-word (BOW). These techniques identify important words in the text collection. However, these approaches have two main drawbacks; (i) ignoring the word order and syntactic structure. (ii) Feature space on which classification models should be trained is sparse and high-dimensional. To address these problems, word embedding techniques have been extensively exploited in recent studies. Mikolov et al. [34] introduced word2vec word representation technique, which captures the semantic and syntactic relationships between words. Word2vec is an estimation-based approach that represents the word in low-dimensional space. Pennington et al. [40] have proposed similar word embedding GloVe that uses occurrence counts of words in the entire corpus during training, instead of fixed-sized window-based method. The main limitation of these word embedding techniques is,(i) limited size of vocabulary and (ii) these word embedding suffers from polysemic problem. To address these aforementioned problems, Bojanowski et al. [7] proposed FastText word embedding technique that trained model with character-level information. In addition to the above mentioned methods, several authors have proposed deep neural networks such as CNNs and RNNs with word embedding for text classification. The reason for this popularity is the ability of CNNs in local pattern and the power of RNNs in sequential modeling. Aslam et al. [3] et al. proposed CNN-based multi-class classifier for automatic classification of

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

user reviews. The authors extracted textual and non-textual information. The authors have evaluated their approach on publicly available dataset of app reviews. They concluded that their proposed approach significantly improved the state of the art with 95%, 93% and 94% of precision, recall and f1-measure, respectively.

Recently, Vaswani et al. [46] introduced a transformer-based architecture for classification that focuses on important parts of a sequence. The transformer architecture consists of an encoder-decoder and a self-attention mechanism. BERT is a novel language model that predicts the sequence of words based only on context. The BERT works in two steps: pre-training and fine-tuning. BERT model can be pre-trained using a source-domain task and fine-tuned by adding different layers to perform various NLP tasks such as question answering and classification. In addition, all pre-training language models have widely established their effectiveness in achieving state-of-the-art performance in a variety of NLP tasks. De Araújo and Marcacini [17] have utilized Requirements Engineering-Bidirectional Encoder Decoder Representation from transformer (RE-BERT) for automatic identification of functional requirements. Their research focused on fine-tuning BERT by capturing the local context between the software requirements. Their proposed approach also identifies software requirements from app reviews. The authors compared traditional BOW techniques with pre-trained language models such as BERT, RoBERT, DistilBERT, and M.DistilBERT for app review classification. They conducted experiments on 3691 app reviews and classified them into four requirement classes. The results showed that pre-trained language models performed better than BoW techniques in app review classification. Yang and Liang [49] have proposed hybrid approach of BERT and topic modeling. They utilize BERT for multi-label classification of app reviews and then implement Latent Dirichlet Allocation (LDA) to extract topic from reviews that help developers to quickly understand the user's requirements.

From the existing literature, we found that exiting approaches use traditional TF-IDF techniques to extract features related to software requirements from user reviews. In addition, few studies have proposed transformer-based models to extract functional requirements from user reviews. To the best of our knowledge, all of these studies have focuses on binary and multi-class classification. Only, Jha and Mahmoud (2019) discussed multi-label user reviews. The authors converted multi-label classifier into a binary classifier using the binary relevance (BR) method. Moreover, their research focused on NFRs classification. However, BR decomposes the multi-label tasks into binary classifiers which increase the computational cost. To alleviate this issue, this study explored the power of BERT model in multi-label review classification. BERT models are extensively used for better feature extraction, because they retain the contextual relationships between words in a sentence. Therefore, this paper investigates the performance of neural language model in multi-label classification problem.

2.1 Multi-label classification

With the rapid development of classification models, multi-label classification models have attracted the attention of researchers. These models are applicable to real world scenarios such as text classification and semantic annotation of multimedia data (e.g., images and videos). Existing single-label techniques become evidently inadequate when an image contains multiple objects or text contain multiple topics in document. Multi-label classification task classifies each data instance into a subset of predefined classes [48]. These multi-label classification models mainly include three categories, i.e., problem transformation-based method, algorithm adaption-based methods and ensemble model-based methods. The first category of this group is problem transformation method. This method transforms multi-label problem into single label problem. The popular examples of this method are Binary Relevance (BR), Classifier Chains (CCs) and Label powerset (LP) [8]. These algorithms transform a multi-label classification problem into series of binary classification problems.

The second group of multi-label classification problem can be called adaptation-based methods. These algorithms enable traditional machine learning algorithms to handle multi-label classification problems. Traditional machine learning algorithms such as NB and SVM are combined with BR strategy results in an algorithm called BR_{SVM} and BR_{NB} [25]. Existing literature shows that other adoption method techniques such as multi-label KNN (ML-kNN) [48] and multi-label decision tree (ML-DTs) [14] are used to handle multi-label classification problem. These techniques relying on maximum a posteriori (MAP) principle which trains models to predict proper labels from labeling information embodied in neighbors. ML-kNN also deals with data imbalance issue by estimating the prior probability of each class label. The basic idea of ML-DTs is to build a decision tree that handles multi-label data, where multi-label entropy is utilized to build the decision tree recursively. The Adaptive Resonance Associative Map (ARAM) technique was proposed by Benites and Sapozhnikova [5] which enables neural networks to deal with multi-label classification problems. Ranking- Support Vector Machine (Rank-SVM) is a machine learning algorithm based on the statistical learning theory of classical SVM. This classifier is used to deal with multi-label data to minimize ranking loss (Elisseeff et al., 2001).

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

The third category is called ensemble methods, where two machine learning techniques (problem transformation methods and adaptation algorithm methods) are combined to improve the performance of classification technique. Random k-Labelsets (Rakel) (Tsoumakas et al., 2011) and Ensemble Classifier chain (ECC) [32] are the popular example of this category. In the Rakel technique, each base classifier is trained on small dataset and then single-label classifier is trained to predict the power set of each random subset. ECC considers CCs as base classifier; the relevant labels are predicted by comparing the final prediction with the threshold value. The existing studies on multi-label classification discussed by Jha and Mahmoud [25] suffer from the following limitations.

- Traditional multi-label classification methods require data preprocessing and transformation, results in increases the computational cost and biasness.
- In order to obtain satisfactory results of multi-label classification problem, there is need for better approach for aforementioned problem [28, 32]

To address the aforementioned limitations, deep neural network-based models are used for multi-label text classification. However, existing deep learning models require large amount of data for training, which is considered as the limitation of model. To alleviate this issue, this research work adopts transfer learning-based that requires small data for model training.

3 Methodology

3.1 Motivation

The prominence of mobile apps has led the development of consolidate relationships between online users and developers. These popular apps allow users to express their reviews and receive millions of reviews written in the natural language. Such reviews contain valuable information for requirements engineers on various aspects of the apps such as functionality, usability and feature requests. Manual classification of relevant reviews is a fierce task [42]. According to prior studies [23] [36], a popular app Facebook receives 4000 reviews on daily basis. However, only 30% reviews contain requirements-related information, whereas 70% contain noisy and irrelevant information. Automatic classification of RE-related information has become the hot topic in field of RE. In context to this, several studies have discussed lexicon-based and machine-based methods for automatically filtering the relevant reviews to support the further app development. Panichella et al., [38] have discussed the importance of NLP and machine learning in detection of relevant and irrelevant reviews. Binkhonain and Zhao [6] conducted systematic literature survey on machine learning techniques for identifying NFRs from formal requirements documents. Although promising results have been reported, many studies require significant human efforts and decrease the overall performance of the model. This fact can even make these techniques are infeasible for scenarios with multi-label reviews. For example, Jha and Mahmoud [25] have extracted NFRs from multi-label dataset and reported poor precision score of 0.66 for Dependability class and 0.59 for performance class. This causes a development team to be overwhelmed with false positive.

To overcome the limitations of Jha and Mahmoud [25], this research paper investigates the effect potential of BERT model in multi-label classification of user reviews. Specifically, we aimed to determine whether BERT is capable of achieving new state-of-the-art results. For this, we compared its performance with approaches developed by Jha and Mahmoud [25], which represent the best performing machine learning technique for this task. Therefore, we used the dataset and annotations of Jha and Mahmoud [25] as the benchmark dataset. We decided to build on this datasets since: (1) they were carefully annotated using peer annual content analysis and contained only user comments for which the annotator agreed on the same label. (2) They serve as a gold-standard corpus and allow us to compare our methods with state-of-the-art methods. An overview of dataset is discussed in detail in Sect. 6.

4 Research gap

Based on the existing comprehensive literature review, some of the vital limitations have been identified.

1. Minimal research has been conducted on the classification of NFRs from app reviews.
2. Most studies have used traditional feature selection methods, such as TF-IDF.
3. No language model has been used benchmark dataset for the prediction of NFRs from app stores.
4. Manual feature selection for NFRs is a laborious and time consuming task and outcries for the use of transfer learning.
5. Surprisingly, RE community has not studied TL-based multi-label classification model to classify user reviews into NFRs.

To bridge these gaps, this proposed research work develops a framework MNoR-BERT framework to classify NFRs for effective decision making. The main objective of this paper is (a) deploy BERT model to classify multi-label user reviews in NFRs. (b) Does application of BERT model in multi-label classification leads to new state-of-the-art

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

results. To evaluate the proposed approach, we present the results of experiments conducted on dataset of 6000 app store reviews collected by Jha and Mahmoud [25]. Based on the datasets we apply our approach on the following tasks.

Task 1 Multi-label classification of four (DE, PE, US, and SU) on all NFRs, excluding Miscellaneous (i.e. 2369 reviews).

Task 2 Multi-label classification of all NFRs in the original dataset, i.e., 6000 reviews.

For the aforementioned tasks, the experiments were designed to answer following research questions:

4.1 RQ1: Does the transformer-based MNoR-BERT approach perform better than Binary relevance for multi-label classification?

The performance of traditional binary relevance for multi-label classification decreases, because it ignores the correlation and dependencies among labels. Therefore, this proposed work aims to study how a transformer-based approach classifies multi-label requirements using BERT word embedding.

4.2 RQ2: Does the transfer learning-based approach perform better than traditional keyword-based approach at classifying app store reviews?

This research question compares the performance of transformer-based classification approach to keyword-based approach to classify app reviews into general NFRs classes. This research paper explored how the pre-trained model performs better than existing technique, i.e., keyword-based approach [25]. The existing research work is based on term matching rules with TF-IDF word representation technique. The results can provide insights into the applicability of using language models.

4.3 RQ3: Does the BERT word representation technique perform better than traditional word representation techniques, such as TF-IDF and GloVe?

The proposed research work compares the accuracy of BERT-based word representation technique to traditional word embedding techniques such as TF-IDF and GloVe. These word representation techniques are compared based on evaluation metrics such as Precision, Recall, f1-measure, accuracy, and Hamming score.

5 Pre-trained language model

In this section, the proposed approach in this paper is introduced in detail The main objective of the pre-training language model is to learn the representation of natural language. Generally, these language models are used to predict the next word from a set of previous words in unidirectional manner. In unsupervised learning, probabilistic language modeling is used to estimate probability density. Using these models, the next word in the sentence is predicted from the set of previous words or contexts. The traditional objective of the language model is to calculate joint probability $P(s)$ given in Eq. (1). For the given text sentence s = (s_1, s_2… s_N). The joint probability can be depicted as follows: [16] (Yang et al., 2019)

$$P(s) = \prod_{k=1}^{N} P(s_k | s_{<k}) \tag{1}$$

Then, conditional probability distribution is evaluated by neural network. The autoregressive forward factorization function (given as below) is used to improve the probability of pre-training.

$$\begin{aligned} max_\theta \log p_\theta(s) &= \sum_{k=1}^{N} \log p_\theta(S_k | S_{<k}) \\ &= \sum_{k=1}^{K} \frac{\exp(h_\theta(s_{1:k-1})^T e(s_k))}{\sum_{s'} \exp(h_\theta(s_{1:k-1})^T e(s'))} \end{aligned} \tag{2}$$

where, $h_\theta(s_{1:k-1})$ is the hidden state vector representation according to the parameter Θ provided by neural network models and e (s_i) indicates the embedding vector of word s_i.

Traditional word embedding techniques such as word2vec, GloVe, ELMo and FastText based on the fact that words in similar context have similar meaning are used to conflate into a single vector representation. These techniques are unable to retain contextual relationships between the words. Due to the dynamic nature of words, their contextual representation of words varies based on their meaning in context. This assumption is considered the main drawback of traditional word embedding techniques. To address this problem, Vaswani et al., [46] introduced a transformer-based architecture to better capture the contextual representation of text. Radford and Salimans, 2018 proposed transformer-based architecture that includes attention layers for NLP tasks such as sequence classification.

The purpose of BERT is to predict the sequence of words based only on their context. BERT model along with encoder-structure is used to solve eleven NLP tasks such as sequence classification, entity recognition and question answering etc. Unlike existing transformers-based

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

architectures that only utilize unidirectional (i.e., left-to-right) architectures, BERT introduces bidirectional pre-training to incorporate text in both directions. In addition, the BERT architecture was designed to enhance the new representation of the input text with contextual information. This pre-trained model consists of multi-headed attention that focuses on different parts of the text and achieves an improvement in performance. By following the same vein, the proposed research work addresses multi-label classification problem with complex word embedding techniques such as BERT, which achieves remarkable performance. The architecture of proposed approach is represented in Fig. 1.

BERT works in two steps, pre-training and Fine-tuning. BERT model was trained on the BooksCorpus (800 million words) and English Wikipedia (2500 million words) and fine-tuned by adding additional output layer in order to achieve state-of-the-art results in downstream tasks. The BERT model is build based on transformer encoder [46]. There are two models, BERT base and BERT Large and their sizes are given below: BERT base (L = 12, H = 768, A = 12, total parameter = 110 M) BERT Large (L = 24, H = 1024, A = 16, total parameter = 340 M). Where, L denotes number of layers (transformer blocks), H is the hidden size and A is attention tasks.

In this paper, BERT utilizes the Masked Language Modeling that masks out 15% from the input sequence and then model is trained to predict masked token from the contextual representation of that sequence. For example, instead of the sentence "my dog is hairy" the input is "my dog is [MASK]"[18]. However, in downstream task mismatch is occurs in pre-training and fine tuning task. To address this issue, Devlin et al. divide the input sequence into 80:10:10 proportions, where 80% tokens are replaced with masked token, 10% tokens are replaced with random token and 10% are original or unchanged tokens of input sequence. BERT model is used to predict the original token. Given a text sequence s, BERT model randomly replaced the words or tokens with [MASK] token. From the following Eq. (3), masked tokens are predicted from original token. Here, S_{masked} represents the masked tokens and $S_{non-masked}$ represents the original tokens.

$$\max_\theta \log p_\theta(S_{\text{masked}}|S_{\text{non-masked}}) \approx \sum_{k=1}^{N} m_k \log p_\theta(S_k|S_{\text{non-masked}}) = \sum_{k=1}^{N} m_k log \frac{\exp\left(H_\theta(S_{\text{non-masked}})_k^T e(s_k)\right)}{\sum_{s'} \exp\left(H_\theta(S_{\text{non-masked}})_k^T e(s')\right)} \quad (3)$$

Here, m_k=1 indicates that s_k is the masked token; H_θ parameter is Transformer that calculate hidden vectors of sentence s of length k with Θ parameter.

6 Multi-label review classification

Let $D = \{(R_i, L_i)\}_{i=1}^{N}$ denotes the dataset, which contains N reviews associated with labels $L = \{0, 1\}^l$, where L is the total number of labels and l represent the presence or absence of label. Let $R_i = \{r_1, r_2, \ldots, r_k, \ldots, r_n\}$ indicate the i-th review contain n words with r_k being the kth word in the review, The multi-label classification task requires training a classifier to assign most relevant label to a review sentence. Furthermore, each label corresponding to the review can be encoded in binary form. Each review was represented with binary representation of list of labels. The list value one indicates that, review belongs to that specific label and zero otherwise. The visual representation of encoded labels corresponding to each review is shown in Table 1.

Due to the informal nature of user reviews, preprocessing steps are followed to convert the input text into an understandable format of pre-trained language. This research paper deals with user reviews written in natural language. Each word in the review statement was converted into token by using BERT tokenizer. After tokenizing each review, padding tokens [PAD] are inserted to adjust the length of all reviews. BERT considers an input sequence with fixed length of maximum tokens, i.e., 512. To choose suitable length, we analyzed the length of user reviews in the datasets. Even with fixed length of 128, we covered more than 99% of user reviews. Thus, user reviews containing more tokens are shortened accordingly. Since, with small amount, only little information is lost. Thus, this paper chose fixed length of 128 tokens instead of 512 tokens to minimize computational requirements. Then, special tokens such as the classification [CLS] the separator [SEP] tokens are inserted to the input sequences. [CLS] token represents the summary of the entire user review and [SEP] tokens are used to differentiate two sentences.

The typical pre-trained BERT model has input layer, hidden layer of size H, pooling layer and output layer. Generally, input and hidden layers are considered as the encoder, and pooling layer and output layers are considered as decoder layer. The BERT takes tokens as input associated with embedding vector of the size H. The first token [CLS] indicates the beginning of each input sequence. The hidden layer corresponding to this essential [CLS] token summarizes the input sequence (r_i) representation for classification purpose, which we represent as $S_i \varepsilon R^H$. Here, S_i indicates the embedding vector of hidden layer corresponding to [CLS] token.

Then, separation token [SEP] is used to differentiate the sentences. In addition, the final representation of the token was obtained by aggregating segment and position embeddings. This output is fed to simple neural network.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

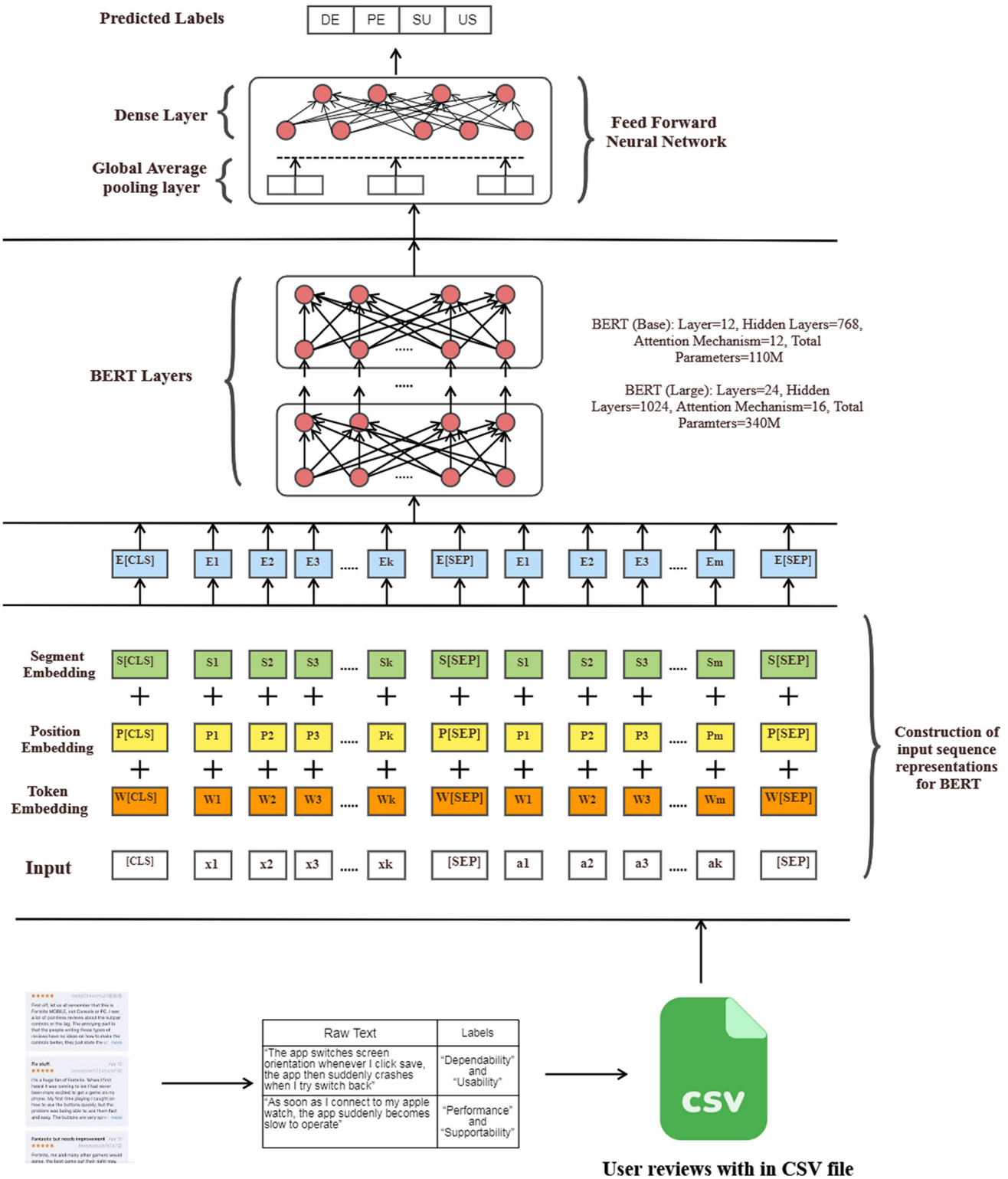


Fig. 1 Architecture of BERT model for non-functional requirements classification

The output is computed as $W\varepsilon R^{l\times H}$, where W is depicted as the weight of classification layer, l is the unique number of labels and H is the size of hidden layer. Then, sigmoid function is used as a output function of the model. The reason to use logistic sigmoid function in this paper is the multi-label nature of the problem. Generally, BERT model has uses the softmax function as activation function, which sums up all the probabilities equal to 1. But, softmax

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 1 Multi-label associated with user reviews

Review no	DE	PE	SU	US
8	0	0	1	1
9	0	0	0	1
25	1	0	0	0
208	1	0	0	1
724	0	1	0	1
793	0	1	1	0
797	1	0	1	0

function is best suitable for single-label classification tasks, where classes are mutually exclusive. On the contrary, sigmoid function considered for multi-label classification where classes are not mutually exclusive. It computes the problem for each label independently, rather than computing on labels. Moreover, binary cross entropy is used as the loss function. The objective of the following formula is to calculate binary cross-entropy:

$$\text{Loss} = -\sum_{i=1}^{n} \widehat{x_i} \log \sigma(x_i) + (1 - \hat{x_i}) \log(i - \sigma(\hat{x_i})) \qquad (4)$$

where, $\sigma(x_i) = 1/1 + e^{-x}$ is the sigmoid function, x_i is the raw output of the fully connected layer.

7 Experiment

In this section, the description of datasets is discussed, followed by the evaluation measures that are used to evaluate the performance of the model on these datasets. The baseline methods along with experimental settings are also explained.

7.1 Dataset

In this research paper, we make use of two different multi-label dataset created by Jha and Mahmoud [25]. The first dataset consists of 6000 user reviews only from Apple Store. On first dataset, Jha and Mahmoud [25] extracted NFRs using binary relevance, and on second dataset they have applied their keyword matching approach to extract NFRs. We utilized this same multi-label dataset for the sake of comparability. We downloaded this dataset from a link provided by Jha and Mahmoud, [25] in their research paper. The dataset contains reviews of 24 iOS apps. These reviews were manually examined by three human annotators to classify them into the NFRs classes.

The second dataset is also collected by Jha and Mahmoud [25]. A direct link of both datasets is available on (http://seel.cse.lsu.edu/data/emse19.zip. This second truth dataset consists of 1100 reviews from 12 different Goggle's app store and Apple's app stores. This dataset contain reviews of iMovie, Weather, Und, TrueCaller, SweatCoin, Transit, Netflix, OneDrive, Amazon, Music, iTunes U, Realtor, Dictionary and Temple Run from Apple's App Store; other reviews were collected from Android's Google Play stores of same aforementioned apps, excluding the apps iMovie and iTunes U since they do not have Android versions.

The authors have developed MARC tool to support judges in order to reduce their classification effort. The dataset was annotated by two PhD students and one undergraduate student of computer science. The 6000 user reviews consists of 2369 non-functional requirements and 3631 of miscellaneous requirements. These label reviews belongs to different domain such as games, communication, books, and health etc. They classified these reviews into four NFRs classes based on the definition of each class, which are (I) Dependability, (II) Performance, (III) Supportability, (IV) Usability. Table 2 shows a sample of review sentence. Table 3 shows the multi-label reviews with sample number, and Table 4 shows the distribution of classes in each category. These requirements are associated with at least one label from the subset of label. Figure 2 shows the graphical representation of two multi-label dataset.

Both datasets come in an Excel file, which contain two attributes. The first one represents the class label as one of the NFRs, and the second one represents the text of the review.

7.2 Evaluation measure

Prior studies on multi-label review classification have used various evaluation metrics for classification results. In multi-label classification problem, one or more labels can be associated to each review. These labels belong to a predefined set of labels. Therefore, multi-label classification problem is typically harder and more complicated than binary classification. From the literature, four standard evaluation measures are adopted for the comparison of classification methods. These measures are extensively used to assess the performance of the models and to find the misclassified instances (false negatives or false positives) that cause high costs. Based on the imbalance nature of dataset used in this paper, the models are not compared on the basis accuracy metrics. We considered Recall, Precision and F1-measure for the comparison. The reason behind the selection of these measures is that high recall value indicates the false positives and low recall values are associated with risk of missing important information about the quality attributes. Consequently, we seek an acceptable balance between precision and recall, so that

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 2 Definition of four review sentences types used in Jha and Mahmoud [25]

Sentence type	Definition	Example
Dependability	Dependability refers to the reliability, availability, and security of the app. In general, user reviews raise concern about the trustworthiness of the app	Can't use the app if it keeps crashing. Ever since the update it crashes. I'm paying for a music service that I can't use. Does the app meet all requirements?
Performance	This category related to user concern on the performance of the app, such as its response time, scalability, and resource consumption	Been trying to use the app for a few days now and it always stalls and freezes. Overall really good app. It's a little slow getting new houses up and on, but only a day or two later. I like it
Supportability	Supportability refers to the reviews discussing update or maintenance issues of the app. Moreover, it includes issues related to connectivity, compatibility, interoperability and portability	Hopefully Apple Watch Support will return. A recent update removed the Apple Watch app. Makes it useless! Doesn't work on my iphone x
Usability	Usability refers to the user reviews related to GUI and user operation on the app. According to taxonomies, the ease-of-use of the app, documentation, understandability, and readability are included	Easier to use than the mobile website. I like the ability to bound searches by circling the desired area on a map. I like the game itself, but the graphics could use some serious work

Table 3 Multi-label (ML) dataset

Number of labels	Number of samples	Sentence type	Associated with
One	2004	The time and date is all wrong. Please fix immediately	Dependability
		Until this last update. Used 80 of my battery in four hours. Not acceptable	Performance
Two	344	Frustratingly slow and freezes on iphone 6. Needs to be fixed immediately	Dependability and Supportability
		I updated my iphone to iOS 11 and now I can't use my headphones to control pandora	Supportability and Usability
Three	22	Extremely slow loading on my iphone the last several days and then skips songs all the time usually half way through one I really like	Dependability, Performance, and Usability
		After the last update, I have not been able to use the app. It always crashes	Dependability, Supportability and Usability
		After the last update I haven't been able to play any music on any of my apple devices through the app. I just keep getting the message that my station is buffering and will be right back… Before now I have never had any problems and have always relied on Pandora for some good music	Performance, Supportability and Usability

users are not overwhelmed by false positive. [28, 32] have adopted four evaluation metrics out of 16 metrics for the comparison of classification methods. These four metrics are grouped into two categories, namely, example and label-based metrics. Example-based metrics are evaluated on examples, where each example is associated with the single label, $y_{i,}$, and $y_i \varepsilon Y$, i.e., set of disjoint labels $Y = \{ y_1, y_2, y_3,\ldots, y_n\}$,where $n > 1$. This paper also evaluates the performance of the proposed classifier with two example based metrics, namely Subset Accuracy (SA) and Hamming Loss (HL) and with one label-based metrics macro-average f1-measure. Moreover, this research work has utilized similar evaluation metrics as previous multi-label classification problem discussed by Jha and Mahmoud [25] for the sake of comparability.

To comprehensively evaluate the effectiveness of the proposed classifier above mentioned metrics are used. In this research work, true positives, false positives, true negatives and false negatives are represented as TP, FP, TN, and FN, respectively. The macro precision, recall and F1-measure [21], Gibaja, 2014) are calculated as follows:

Subset accuracy (SA): The subset accuracy is defined as the ratio of correct predictions to the total number reviews in sample.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 4 The number of NFRs per category detected in the reviews sampled from each category

App name	Domain	Dependability	Performance	Supportability	Usability
AccWeather	Weather	35	0	6	69
Adobe Acrobat	Business	1	1	24	71
Avira Antivirus	Utilities	21	4	14	55
FitBit	Health and Fitness	27	5	60	82
Google Maps	Navigation	53	3	14	79
Google Translate	Reference	41	4	24	49
KeepSafe	Photos and Videos	29	1	10	29
Lumosity	Education	2	1	7	21
Pandora	Music	20	8	14	36
PokemonGo	Games	70	5	16	51
Prisma	Entertainment	79	7	3	46
Zillow	Lifestyle	61	4	5	61
7MinuteWorkout	Health and Fitness	4	28	102	11
Boating USA	Navigation	25	2	30	45
Enlight	Photos &Videos	19	2	11	32
HotSchedule	Business	148	4	14	40
Jukebox	Music	21	2	20	47
My Babys Beat	Lifestyle	47	0	0	82
NOAARadarPro	Weather	37	12	4	21
ProCreate	Entertainment	86	5	21	29
StarWalk	Education	39	4	22	25
Super Mario Run	Games	23	4	23	44
Swype	Utilities	106	4	35	59
WolframAlpha	Reference	39	5	14	34
TOTAL		1033	114	493	1118

$$\text{Subset Accuracy} = \frac{\text{number of accurately classified samples}}{\text{total samples}} \tag{5}$$

Hamming score (HS): Hamming score is the ratio of correctly predicted labels to the total number of labels.

$$\text{HS} = \frac{1}{n}\sum_{I=1}^{n}\frac{\left|\hat{Y}_i \cap Y_i\right|}{\left|\hat{Y}_i \cup Y_i\right|} \tag{6}$$

where, $\widehat{Y}_i$ are the predicted labels, and Y_i are the actual label.

Precision (P): Precision reflects as the ratio of number of correctly classified reviews under specific label (TP) to the sum of classified reviews under the same label.

$$\text{MacroPrecision} = \frac{1}{n}\sum_{i=1}^{m}\frac{TP_i}{TP_i + FP_i} \tag{7}$$

Here, n denotes the total number of data sample.

Recall (R): Recall is calculated as the average of number of correctly classified reviews to the total number of reviews belonging to that label.

$$MacroRecall = \frac{1}{n}\sum_{i=1}^{m}\frac{TP_i}{TP_i + FN_i} \tag{8}$$

F-measure: F-Measure is defined as the harmonic mean of precision and recall.

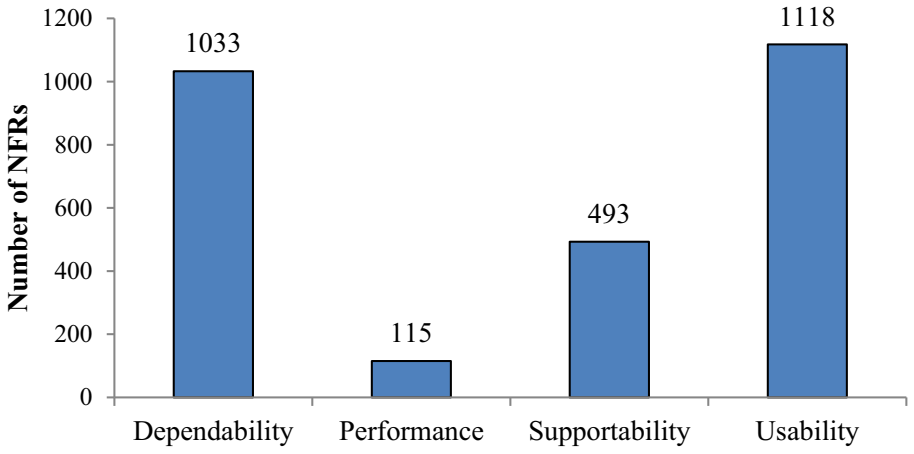
$$F1 - \text{Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

Hamming loss (HL): Hamming loss measures the prediction error, i.e., an incorrectly predicted labels. HL is the fraction of labels, whose correct label is not predicted, and the labels that are incorrectly predicted, over the total number of labels. The value of HL ranges between 0 and 1. The smaller the value of HL indicates that algorithm performs better in classification.

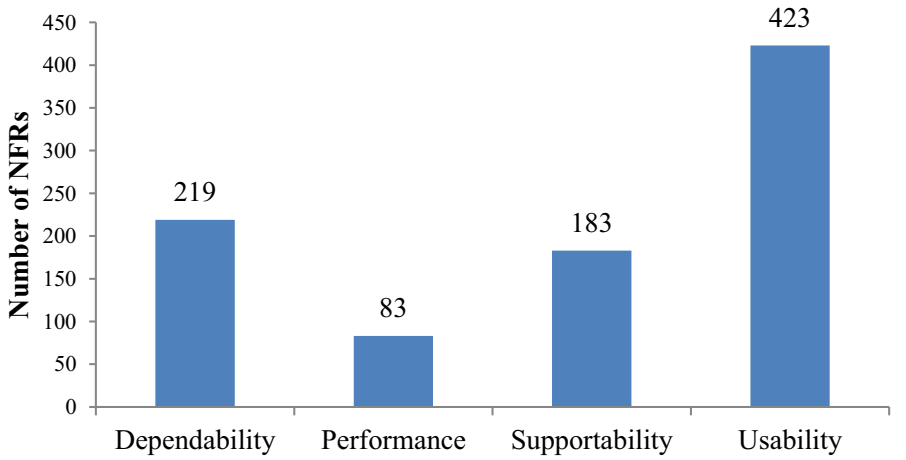
$$\text{Hamming Loss} = \frac{1}{|n|}\sum_{i=1}^{|n|}\frac{x \text{ or } \left(\hat{Y}_i, Y_i\right)}{|l|} \tag{10}$$

where, l denotes the total number of labels, and $\widehat{Y}_i$ indicates the predicated labels and Y_i denotes the real label and xor indicates the XOR operation.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.



(a) First Dataset consists of 6,000 user reviews of iOS store



(b) Second consists of 1,100 reviews from both iOS store and Google Store

Fig. 2 the distribution of NFRs classes in both datasets **a** first dataset of 6000 reviews from iOS apps **b** second dataset of 1100 reviews from iOS apps and android apps

7.3 Experimental setups

In this section we discuss the different system implementations and experimental setup of our research work.

In our technical setup, all the experiments were carried out in Jupyter notebooks using Keras, Scikit-learn, and Tensorflow libraries. For machine learning techniques, Scikit-learn Python library is used for classification problems (Pedregosa et al., 2015). Keras, Tensorflow and Tensorflow_hub are Python libraries used to import the deep learning layers (Chollet et al., 2015). To construct the input matrix, preprocessing steps such as tokenization, removal of stop-words, and lemmatization are applied to datasets. After converting the tokens into vectors, input sequence is padded or truncated to maximal sequence length of 128 for the models. For machine learning techniques, CountVectorizer class is used to convert text into vectors, and GloVe word vectors are used in deep learning models. Figure 3 (a) (b), (c), and (d) show the layers of the deep learning models and their corresponding parameters. In CNN model, filter value and kernel are set to 100 and 3, respectively. In LSTM model, memory unit is set to 100, and in BiLSTM memory unit is set to 100.

After inputting the vector text into embedding layer, pooling layer is used only in CNN and BERT models reduced the dimensionality of vector space generated by the embedding layer. Finally, dense layer as an output layer is stacked with sigmoid function for the classification purpose, and perform multi-label classification of user reviews. The proposed model utilized output layer as the classification head. In Tables 5, 6, and 7, we describe the hyperparameter tuning of deep learning models and BERT model, respectively. The description of four and five classes is discussed in Sect. 3.2.

An Intel Xeon E5-2690 CPU with 16 GB of memory was used in the Microsoft windows Server 2012 R2 communication systems.

In addition, we have used two different pre-trained models base and large model both in cased version. We also investigate the effect of early stopping with 6 epochs with patience value of three avoid the overfitting. The dataset is split into training and testing set with 70:30 ratio,

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 128, 100)	402900
conv1d_1 (Conv1D)	(None, 128, 100)	30100
global_average_pooling1d_1 (	(None, 100)	0
dense_2 (Dense)	(None, 20)	2020
dropout (Dropout)	(None, 20)	0
dense_3 (Dense)	(None, 4)	84

Total params: 435,104
Trainable params: 435,104
Non-trainable params: 0

(a)

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 128, 100)	100000
lstm (LSTM)	(None, 100)	80400
dense_4 (Dense)	(None, 20)	2020
dropout_1 (Dropout)	(None, 20)	0
dense_5 (Dense)	(None, 4)	84

Total params: 182,504
Trainable params: 182,504
Non-trainable params: 0

(b)

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 128, 100)	100000
lstm (LSTM)	(None, 100)	80400
dense_4 (Dense)	(None, 20)	2020
dropout_1 (Dropout)	(None, 20)	0
dense_5 (Dense)	(None, 4)	84

Total params: 182,504
Trainable params: 182,504
Non-trainable params: 0

(c)

Layer (type)	Output Shape	Param #	Connected to
input_word_ids (InputLayer)	[(None, 128)]	0	[]
input_mask (InputLayer)	[(None, 128)]	0	[]
segment_ids (InputLayer)	[(None, 128)]	0	[]
keras_layer (KerasLayer)	[(None, 1024), (None, 128, 1024)]	333579265	['input_word_ids[0][0]', 'input_mask[0][0]', 'segment_ids[0][0]']
global_average_pooling1d (GlobalAveragePooling1D)	(None, 1024)	0	['keras_layer[0][1]']
dropout (Dropout)	(None, 1024)	0	['global_average_pooling1d[0][0]']
dense_output (Dense)	(None, 5)	5125	['dropout[0][0]']

Total params: 333,584,390
Trainable params: 333,584,389
Non-trainable params: 1

(d)

Fig. 3 model summary of all models **a** CNN **b** LSTM **c** BiLSTM **d** BERT

Table 5 Hyperparameters of CNN with different number of classes on the ML dataset

Model	Class	Batch size	No. of epochs	Dense layer	Dropout	Activation function	Output layer unit
CNN + GloVe	4 Classes	16	50	1 Dense 20	0.2	Sigmoid	4
CNN + GloVe	5 Classes	16	50	1 Dense 20	0.2	Sigmoid	5

Table 6 Hyperparameters of LSTM and BiLSTM for different number of classes on the ML dataset

Model	Class	Batch size	No. of epochs	Dense layer	Dropout	Activation function	Output layer unit
LSTM + GloVe	4 classes	16	20	1 Dense 20	0.2	Sigmoid	4
BiLSTM + GloVe	4 classes	16	20	1 Dense 30	0.2	Sigmoid	4
LSTM + GloVe	5 classes	16	20	1 Dense 20	0.2	Sigmoid	5
BiLSTM + GloVe	5 classes	16	20	1 Dense 30	0.2	Sigmoid	5

which indicates that model is trained on 70% set and tested on 30%. The evolution metrics are generated using Scikit in-build library. Table 8 shows the basic tuned hyperparameter used by BERT model in this paper. The proposed research work compared with baseline with the help of evaluation metrics.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 7 Hyperparameters of MNoR-BERT for ML datasets

Model	Class	Batch size	No. of epochs	Dense layer	Dropout	Activation function	Output layer unit
BERT-base	4 classes	16	6	1 Dense 20	0.2	Sigmoid	4
BERT-large	4 classes	16	6	1 Dense 30	0.2	Sigmoid	4
BERT-base	5 classes	16	6	1 Dense 20	0.2	Sigmoid	5
BERT-large	5 classes	16	6	1 Dense 30	0.2	Sigmoid	5

7.4 Cross-validation

In order to carry out the effective classification results of experiments, dataset is randomly split into training, validation and testing set. The random and shuffle split of data sample cannot be considered as representative of whole data. To address this issue, cross-validation techniques are used to check the robustness and validity of model on all data instances.

In cross-validation techniques, a given dataset is divided into k-folds, and each fold split into training, validation and testing folds. Then, the model is trained on training folds, hyperparameters are tuned on validation folds, and finally final results are obtained on test folds with best parameters. The reason to apply cross-validation is to check the performance of classifier on unseen data, also to ensure the credibility of the model on small dataset.

In case of imbalanced datasets, as in our case, stratified cross-validation method is best suited method, where whole dataset is maintained by general proportions, and results in improvement in bias and variance of cross-validation. However, these techniques are inadequate and complicated to multi-label classification problems.

For multi-label classification problem, Sechidis et al., [43] have introduced IterativeStratification cross-validation technique. The algorithm first selects the desired number of data points from each subset of fold and then selects desired number of data points associated with the label. Then, it chooses the label with fewer data points to ensure they are well distributed and then, tie is broken by selecting the largest number of desired data points randomly.

To test the accuracy of proposed approach on multi-label classification problem, this paper used 3 $\times$2 repeated stratified-cross-validation whose code is publicly available and implemented by Trent J. Bradberry.[2] This research work has used stratified cross-validation, instead of commonly used in existing studies, i.e., tenfold cross-validation, because these methods are more suitable for single-label classification problems, also suffer from extensive computational cost when dealing with multi-label classification. The stratification is performed on training set and test set. Then, results obtained from each fold and average over 6.

7.5 Baseline methods

From the prior studies, various adaptation methods have been adopted for multi-label classification problems. In this paper, BR_{SVM}, BR_{NB}, and BR_{KNN} methods enabled the multi-label classification nature of the base classifier. Comparison with other classification methods is out of scope of this paper. The proposed research work selected linear-based classifiers such as Multinomial NB, SVM, DT, KNN, and DNNs (CNN, LSTM and BiLSTM). This research work compared four baseline machine learning models and three deep learning models with MNoR-BERT for multi-label classification problems. The details of these methods are as follows:

- *Binary Relevance (BR)* BR is a transformation technique that produces a binary classifier for each label of the original dataset.
- BR_{SVM} [25]: SVM is a supervised machine learning algorithm used for classification and regression analysis. With the help of BR multi-label classifier, SVM can be used for multi-label classification.
- BR_{NB} [25]: NB is a linear probabilistic classifier based on the bayes theorem. This assumes conditional independence between the attribute values for each class. BR_{NB} indicates that initialize binary relevance classifier with naïve bayes.
- BR_{DT} adopts decision tree techniques to process multi-label dataset.
- BR_{KNN} adopts k-nearest neighbor to classify multi-label data.
- *Convolutional Neural Network (CNN)* This model consists of Convolutional layer, pooling layer and fully connected layer designed for multi-label classification problem. The word vectors are fed into convolutional layer to extract local features and then pooling layer captures the most significant features. Finally, number of nodes is the number of labels at the output layer, and sigmoid function is used to predict the probability of

[2] https://github.com/trent-b/iterative-stratification.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 8 indicates the other hyperparameter of MNoR-BERT

Hyperparameter	Value
BERT-base	L = 12, H = 786, A = 12, Parameter = 110 M
BERT-large	L = 24, H = 1024, A = 16, Parameter = 340 M
Learning rate	1e-05

classes. The summary of the CNN model is shown in Fig. 3.

- *Long-Short term Memory (LSTM)* In this model, one layer with LSTM is applied to capture the long dependencies, and results are fed into feed forward network with one hidden layer to predict NFRs categories. The summary of the LSTM model is shown in Fig. 4.
- *Bidirectional LSTM (BiLSTM)* In this model, one layer with bidirectional LSTM is applied to word vector. The key difference between LSTM and BiLSTM is that BiLSTM considers both proceeding and succeeding contexts. Then, the output of BiLSTM is fed to linear decoder to predict multi-label NFRs.
- *MNoR-BERT (proposed model)*: this model represents the multi-label classification problem as neural network with multiple output nodes. Each label in the dataset corresponds to an output node of the network, and the output layer can model the dependencies between different categories.

8 Results and discussion and threats to validity

This section presents the experimental results for dataset consists of 6000 user reviews in Table 9. The proposed research work investigates the performance of language model BERT on multi-label classification and compared its performance with existing techniques. The results demonstrate that BERT can easily distinguish the features and classify multi-label sentences more efficiently than binary relevance.

From the results reported in Table 9, we can empirically prove that proposed model performs better than prior techniques, with higher macro F1-measure. It is also demonstrated that BERT model efficiently captures the requirements features compared to keyword and TF-IDF methods. The proposed model yields the highest value of macro F1 of 0.74. In addition, evaluation metrics such as precision and recall is 0.79 and 0.65, respectively. Among the baseline models, proposed model outperforms in precision, recall, f1-measure and accuracy.

8.1 RQ1 Does the transformer-based MNoR-BERT approach performs better than binary relevance for multi-label classification?

Binary relevance transforms the multi-label classification problem into multiple independent binary classification problems, where each binary classification problem corresponds to a possible label in the label space. However, it ignores the correlation and dependencies among the labels. In addition, BR may increase computational complexity of the classifier for large dataset. This research paper compares the binary relevance machine learning with pre-trained language model for multi-label classification. From Table 9 techniques discussed in Group A and B have used preprocessing techniques to remove stop words, to convert lower case letters and perform tokenization. However, no pre-processing techniques are applied in BERT models discussed in Group C. The performance comparison of MNoR-BERT against other models is provided in Table 9, where the models are categorized into three categories.

- **Group A**: include traditional machine learning models.

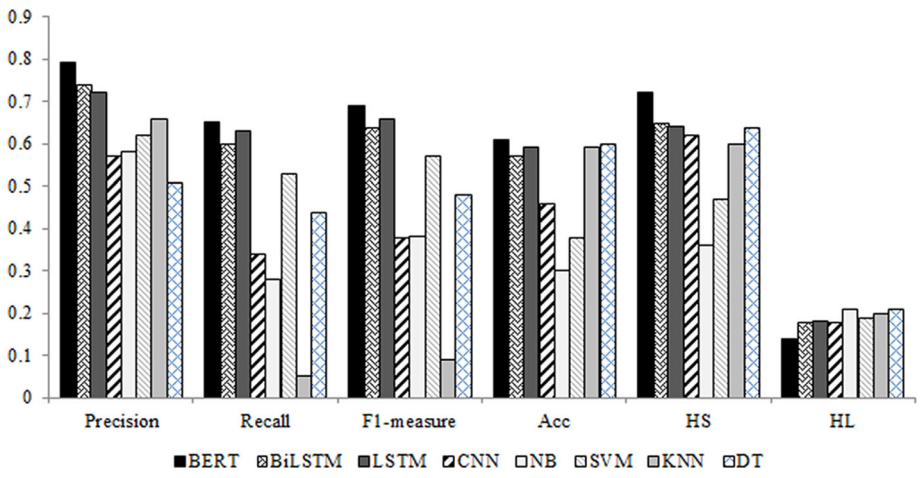


Fig. 4 Comparison of the obtained results of average of all classes

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 9 The comparison of results obtained using proposed approach and the adaption algorithms. *P*: Precision, *R:* Recall, F1: F1-measure

Group	Configuration settings	Dependability	Performance	Supportability	Usability									
		P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	Average F1
A	$BR_{NB+STM+SW}$ [25]	0.68	0.53	0.59	0.25	0.03	0.12	0.71	0.23	0.34	0.68	0.35	0.46	0.38
	$BR_{NB+STM+SW+D}$ [25]	0.69	0.53	0.60	0.25	0.03	0.05	0.69	0.26	0.40	0.67	0.38	0.46	0.38
	$BR_{SVM+STM+SW}$ [25]	0.66	0.56	0.60	0.65	0.59	0.62	0.59	0.48	0.53	0.56	0.46	0.50	0.56
	$BR_{SVM+STM+SW+D}$ [25]	0.67	0.58	0.62	0.69	0.62	0.65	0.62	0.52	0.56	0.56	0.45	0.50	0.58
	$BR_{KNN+STM+SW}$ (Our Implementation)	0.34	0.03	0.05	1.00	0.03	0.05	0.80	0.12	0.20	0.51	0.04	0.07	0.09
	$BR_{DT+STM+SW}$ (Our Implementation)	0.58	0.49	0.53	0.57	0.50	0.53	0.49	0.39	0.43	0.42	0.38	0.40	0.47
B	CNN + GloVe	0.75	0.67	0.71	0	0	0	0.82	0.08	0.14	0.71	0.63	0.67	0.38
	LSTM + GloVe	0.76	0.79	0.78	0.78	0.37	0.50	0.61	0.62	0.61	0.75	0.74	0.74	0.68
	BiLSTM + GloVe	0.72	0.80	0.76	0.90	0.39	0.55	0.65	0.50	0.57	0.76	0.72	0.74	0.65
	CNN + GloVe (Mutli_all)	0.79	0.40	0.53	0	0	0	0.82	0.16	0.26	0.54	0.13	0.21	0.25
	LSTM + GloVe(Multi_all)	0.62	0.58	0.60	0	0	0	0.71	0.34	0.46	0.58	0.38	0.46	0.59
	BiLSTM + GloVe (Multi_all)	0.63	0.62	0.62	0.83	0.29	0.43	**0.83**	0.36	0.50	0.51	0.56	0.53	0.52
C	BERT (base) (70:30)	**0.81**	0.80	**0.80**	0.91	0.40	0.56	0.70	0.64	**0.67**	0.73	**0.77**	**0.75**	0.69
	BERT(Large) (70:30)	0.75	**0.85**	0.80	**0.94**	**0.60**	**0.73**	0.65	**0.68**	0.66	**0.79**	0.76	**0.78**	**0.74**
	BERT (base) (Multi_all)	0.76	0.68	0.72	0.88	0.60	0.71	0.69	0.64	0.66	0.63	0.53	0.58	0.68
	BERT (Large) (Multi_all)	0.69	0.76	0.72	0.76	0.59	0.67	0.67	0.51	0.58	0.74	0.48	0.58	0.64
Cross-validation (Repeated multi-label)	BERT (Base)	0.74	0.79	0.76	0.77	0.59	0.60	0.67	0.70	0.67	0.75	0.74	0.67	0.67
	BERT (Large)	0.75	0.80	0.77	0.78	0.54	0.63	0.63	0.69	0.76	0.76	0.77	0.76	0.73

Bold values represent the good performance of the model

Springer

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

- **Group B:** includes deep learning models.
- **Group C:** includes BERT (Base and Large model)

It is worth mentioning that the results of existing techniques discussed in Group A are reproduced with the help of python code available in their original paper. In addition, it can be seen that the proposed approach shows promising results across individual classes. We achieved accuracy of 61% and f1-measure of 69%, which is considerable improvement over the results obtained by Jha and Mahmoud [25]. From these results, we can conclude that the transfer learning-based approach is quite effective when dealing with the multi-label classification and extract distinctive features to classify the reviews into specified classes. This research question evaluates the performance of transfer learning in multi-label classification. The generalizability of the approach can also be validated by using cross-validation stratified techniques. For this aforementioned purpose, the proposed research work applied three-fold repeated multi-label stratified. Therefore, Table 9 indicates that proposed model also performs better in complex repeated multi-label stratified configurations. This research question also compares the performance of both BERT models such as large and small in cased versions. To measure the effect of both versions, a set of experiments are conducted. From Table 9, it can be noted that the results on BERT large with cased version perform similar or better than the small cased. In addition, the average of all the classes is represented in Fig. 4.

We were able to achieve 85% of F1 score and 0.08 hamming loss on test set, which is considerable improvement over results obtained by previous studies. For instance, Jha and Mahmoud [25] achieved F1-score of 78% on their test set. These results indicate that our transfer learning-based approach is quite effective when dealing with the multi-label classification and extracting distinct features to classify user reviews. Table 10 also indicates that dictionary based approach suffers from false positive values whereas proposed approach has significant balance between precision and recall. In addition, cross-validation techniques are used to demonstrate the generalizability of proposed approach. The results indicate that proposed approach performs better in complex stratification technique. In Table 10, the results in first five rows are reported by Jha and Mahmoud [25], and next four rows are the results obtained from the proposed approach on same dataset with train_test_split (6–7 rows) and cross-validation technique (8–9 rows).

8.2 RQ3 Does BERT word representation technique performs better than traditional word representation techniques such as TF-IDF and GloVe?

RQ3 is concerned with the performance comparison of three word representation models such as TF-IDF, GloVe, and BERT pre-trained language model. The first model used TF-IDF, second model use GloVe word representation with neural network, and third model used BERT language model with fine-tuning. On average, the BERT language model achieved better performances than GloVe and TF-IDF in terms of precision, f1-score and hamming score (shown in Fig. 5). The reported average recall value of BERT is less than TF-IDF. From Fig. 5, the high value of recall and low value of precision indicate that there is more false positive value. However, BERT performs better in terms of precision and recall. There is a reasonable difference between precision and recall. From Fig. 5, we can demonstrate that for each class BERT performed better than GloVe and TF-IDF. In relation to RQ3, we conclude that BERT model that does not require manual pre-processing achieves better results in multi-label classification.

8.3 Discussion

Based on the previous results in Sect. 7.1, the analysis revealed that BERT model outperforms other techniques. For example, Jha and Mahmoud, [25] achieved poor precision value of 0.25 for performance and 0.51 for usability while categorizing NFRs. This indicates that their proposed approach overwhelmed development team with more false positive values. However, the proposed MNoR-BERT approach considerably improves the accuracy of multi-label classification problem. The key difference between MNoR-BERT and previous studies is that it uses BERT as a contextual representation technique that retains the sequence of context vectors along with their semantic meaning. As we can see from the Table 9, the transfer learning-based proposed model reports remarkable results on multi-label dataset. BERT model achieved better subset accuracy of 85% than neural networks with 68% accuracy. The lowest value of Hamming Loss and higher values of precision, recall and F1-measure indicate empirical evidence of competitive performance of proposed model. Both models (Base and Large) significantly outperformed the other models. The results prove that transfer learning models can be utilizes for multi-label classification, and

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 10 The comparison of results obtained by traditional keyword-based approach and proposed approach. TM: Term Matching, TF: Term Frequency, IDF: Inverse Document Frequency

Classification Settings	Dependability	Performance	Supportability	Usability	Average score													
	P	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1	HS	SA	HL	*P*	*R*	F_1
TM	0.60	0.90	0.73	0.39	0.98	0.55	0.45	0.99	0.62	0.60	0.85	0.70	0.60	0.48	0.19	0.51	0.93	0.65
TM + 12	0.76	0.75	0.75	0.74	0.74	0.74	0.67	0.95	0.79	0.69	0.70	0.69	0.69	0.60	0.12	0.72	0.79	0.75
TM + TF-IDF + (12)	0.70	0.78	0.74	0.48	0.91	0.63	0.60	0.97	0.74	0.66	0.72	0.69	0.66	0.55	0.14	0.61	0.85	0.70
TM + TF + (12)	0.72	0.81	0.76	0.66	0.93	0.77	0.66	0.98	0.79	0.67	0.75	0.70	0.69	0.61	0.12	0.68	0.87	0.76
TM + TF + (12) + Android	0.74	0.89	0.81	0.71	0.88	0.78	0.66	0.92	0.77	0.80	0.69	0.74	0.74	0.69	0.10	0.72	0.85	0.78
$BERT_{cased}$ (Base) (70:30)	0.89	0.84	0.86	1.00	0.58	0.74	0.84	0.90	0.87	0.83	0.98	0.90	0.85	0.81	0.08	0.89	0.83	0.84
$BERT_{cased}$ (Large) (70:30)	**0.94**	**0.89**	**0.92**	**1.00**	0.67	**0.80**	**0.85**	0.88	**0.87**	**0.89**	**0.98**	**0.94**	**0.88**	**0.84**	**0.06**	**0.92**	0.86	**0.88**
Repeated multi-label stratified																		
$BERT_{cased}$ (Base)	0.84	0.86	0.84	0.99	0.54	0.68	0.83	0.78	0.80	0.81	0.93	0.90	0.83	0.75	0.09	0.88	0.78	0.81
$BERT_{cased}$ (Large)	0.85	0.84	0.84	0.90	0.70	0.79	0.86	0.86	0.86	0.89	0.92	0.90	0.80	0.78	0.08	0.88	0.83	0.85

Bold values represent the highest precision, recall achieved by the proposed model

they outperforms deep learning and machine learning models (**RQ1**). The differences between the performances of MNoR-BERT and other techniques are highlighted in Fig. 6. The reason behind performance comparison based on f1-measure and recall is because of imbalance nature of dataset. While interpreting the metrics, we cannot rely on subset accuracy. For that we compare on the basis of f1-measure because f1-score is the harmonic mean of precision and recall. Recall metrics are more important in RE, because it is easier for the developers to discard false positive than manually detect false negative. Furthermore, in **(RQ2)** this research work demonstrates that BERT pre-trained model also performs better than traditional dictionary based approach. Jha and Mahmoud, [25] have employed dictionary based approach that includes distinctive indicator words with their labels. However, identifying NFRs from keywords requires human intervention besides costly and time-consuming. From Table 9, it can be concluded that BERT model can be applied for better feature engineering. In conclusion, our proposed approach automatically extracts NFRs-related features and reduces the cost and time.

The rationale behind the better performance of the proposed approach compared to state-of-the-art and other deep learning algorithms is that because BERT pre-trained model retains the contextual relationship between the sentences. To show the performance of the proposed approach on four classes: Dependability, Performance, Supportability, and Usability, we have shown the obtained true positive (TP), false positive (FP), true negative (TN), and false negative (FN) for multi-label dataset. Table 9 shows that Binary Relevance suffers from more false positive values than BERT.

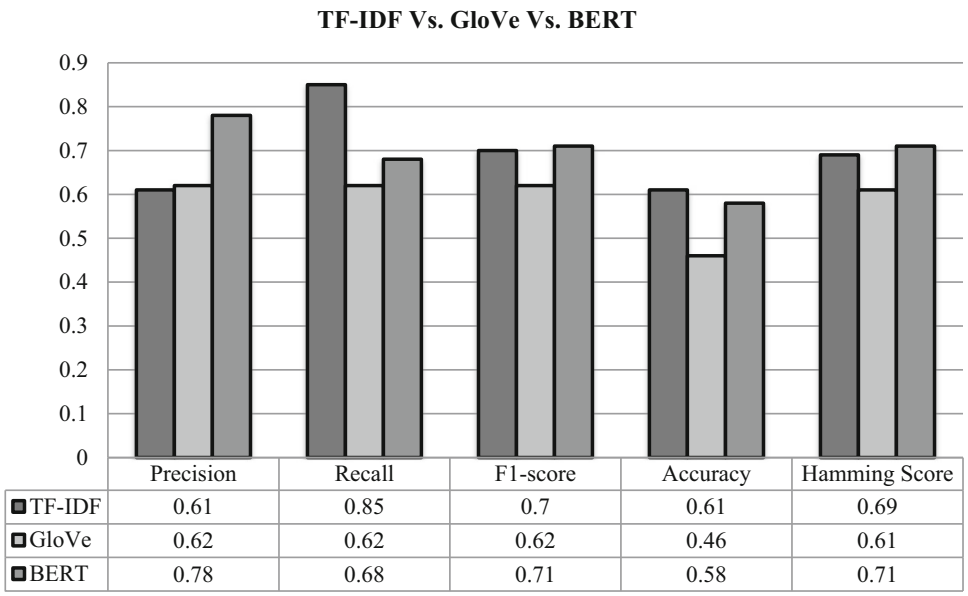
8.3.1 Main finding

Extensive experiments exhibits that our proposed transfer learning-based model in multi-label classification performs better than deep learning and state-of-the-art multi-label classification methods. In all the datasets, BERT outperformed in accuracy, hamming loss, and macro F1 (Table 11).

9 Research and practical impacts

Online user reviews are rich source of information for both users and developers. This research work indicates that transfer learning-based multi-label classification models can be utilized to extract quality requirements also considered as NFRs such as Dependability, Performance, Supportability, and Usability. Due to the rapid growth in feedback provided by online platforms, and its dynamic

Content courtesy of Springer Nature, terms of use apply. Rights reserved.



	Precision	Recall	F1-score	Accuracy	Hamming Score
TF-IDF	0.61	0.85	0.7	0.61	0.69
GloVe	0.62	0.62	0.62	0.46	0.61
BERT	0.78	0.68	0.71	0.58	0.71

Fig. 5 Comparison of results obtained by three word representation technique

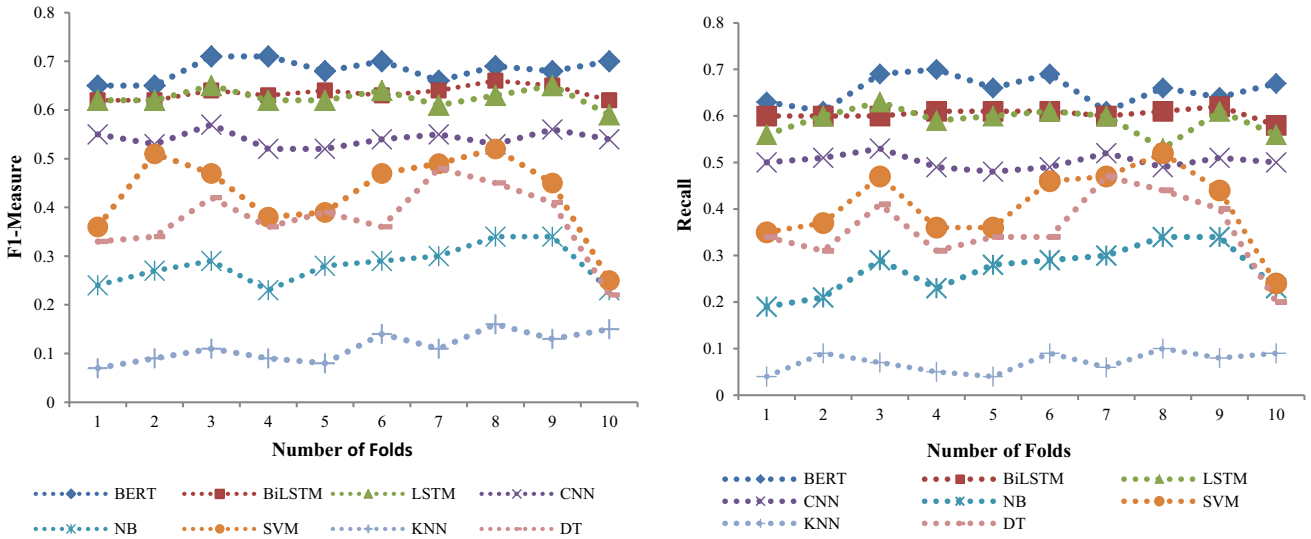


Fig. 6 Performance of BERT vs. other techniques on the basis of F1-measure and Recall

nature, relaying on transfer learning models with additional multi-label classification models can be good option. Thus, the dictionary-based and keyword-based approaches may not always work; because of the informal structure and grammatical errors present in user reviews. The proposed research work automatically classifies multi-label software requirements without using preprocessing techniques, making it more capable of RE practices. Automatic classification of quality attribute into respective classes can improve the software quality. Therefore, NFRs-related information can be used for next release planning. To standout in market competition, mobile app developers must consider the important features for customers identified as needs, expectations, interests and requirements. Recent studies have reported that these reviews can be seen from two perspectives. First, from the users' perspective, such feedback influences their initial thoughts on whether the app is worth purchasing. From the perspective of app developers, user feedback contains technical information that can be useful for planning new features for next release. Positive user reviews attract more customers and bring financial gains, which in turns analyzed to develop better application updates. Despite from this, negative reviews can potentially be misleading and often causes sales losses. Such information can be easily utilized by developers who just initiating their business.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 11 The obtained True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) of the proposed MNoR-BERT and Binary Relevance on ML dataset

Algorithm	BERT	Binary relevance												
	TP	FP	FN	TN	P	R	F1	TP	FP	FN	TN	P	R	F1
Dependability	1423	**57**	123	188	0.96	0.92	**0.94**	1251	**238**	252	59	0.84	0.83	0.83
Performance	1762	**6**	20	12	0.99	0.99	**0.99**	1758	**10**	13	19	0.99	0.99	0.99
Supportability	1552	82	51	115	0.95	0.97	**0.96**	1582	**52**	87	79	0.97	0.95	0.96
Usability	1344	**99**	175	82	0.93	0.88	**0.90**	1313	**130**	190	167	0.90	0.87	0.88

Bold values represent the model that predicts more true positive values

9.1 Comparison with related work

Genc-Nayebi and Abran [20] demonstrates that identification and categorization of reviews is positioned before maintenance and release planning. This means review identification and categorization are very important in research direction to reduce the gap between app reviews and software requirements. Among existing related works, researches conducted by Jha and Mahmoud, [25] and De Araújo and Marcacini [17] are the most similar one to ours.

Jha and Mahmoud [25] proposed a dictionary-based classification approach. They used user reviews collected from iOS and Google Play stores. Their classification recalls are between 23 and 59% and precisions are between 56 and 64%. The authors have applied Binary Relevance for multi-label classification, and their classification results are much less than ours (i.e., recalls are between 47 and 85%, and precisions are between 75 and 95% on average for all datasets) make sense. Our research work considered same classes as those in Jha and Mahmoud, [25]. In contrast to, Jha and Mahmoud, [25], who applied TF-IDF word representation technique and keyword-based approach in processing user reviews, we proposed pre-training based deep neural network model for multi-label classification. We have applied pre-trained language model in multi-label classification problem because TF-IDF ignores word order and syntactic structure, whereas the BERT language model retains the contextual relationship between the words in sentences. Groen et al., [23] have also extract NFRs from the online user reviews. They used linguistic patterns to identify the requirements, and reported high precision but low recall values. They discussed six NFRs such as usability, reliability, portability, compatibility, performance, and security. However, NFRs should also be considered as these requirements play a vital role in the success of apps. The authors used manual tagging procedure to identify the reviews and found that users mainly write about NFRs. To overcome this manual tagging limitation of existing techniques in context of NFRs, we developed a novel framework that automatically extracts keywords from user reviews and classifies them into specific requirements. In our research work, we used four NFRs as described by Jha and Mahmoud [25]. In conclusion, our proposed method automatically identifies and classifies NFRs from online reviews with balanced precision and recall values.

De Araújo and Marcacini [17] have classified app reviews into bug report, feature request and user experience. The authors compared BoW and pre-trained language models for textual representations. They have performed various experiments to investigate the effects of classical BoW approaches on the most recent neural language models. From their reported results, it can be concluded that neural language model obtained better F1-score than BoW. De Araújo and Marcacini [17] al. have proposed RE-BERT approach for extraction and classification of software requirements. They have introduced two approaches, i.e., exact matching and partial matching. The reported results indicate that exact matching with BERT performs significantly better than partial matching. The proposed research work utilizes BERT for NFRs classification. However, one of the important factors that leads to difference between Araujo and ours is the difference in the dataset. They have implemented their approach on multi-class classification,however, our research work focuses on multi-label multi-class classification. They identified and classified functional requirements form app reviews, whereas this paper demonstrate the classification of NFRs.

9.2 Threats to validity

9.2.1 Internal validity

A potential threat to internal validity of proposed approach is the experimental dataset used in this paper. The human annotator created this dataset with the approach in mind. Therefore, there might be the risk of bias. We mitigated this threat by applying repeated multi-label stratified k-fold cross-validation to avoid biasness and tested the model on data samples that had not been revealed during training. We applied repeated multi-label stratified cross-validation technique. However, the reviews might not be general

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

representative of all kind of reviews. Moreover, the datasets that we used have some further issues regarding internal validity. Some reviews may be incorrectly labeled and selection of reviews for the dataset is biased. This results in imbalanced datasets that miss out on some kinds of reviews, for example, Performance. In addition, other major internal threat to the proposed is the dataset. The reviews were encoded by PhD students and thus might not illustrate industry standards. Therefore, conclusions regarding the generalizability based on this dataset are not warranted. We used these datasets as they are used in previous published research work and has its validity and allow us to directly compare our results to existing research work.

Another threat to internal validity is tuning of the hyperparameter. The BERT model has large number of hyperparameters that can be tuned to improve the performance. These sets of hyperparameters include attention mechanism, number of hidden layers, batch size, and dropout size. Tuning these hyperparameters infeasible in practice because, the computational cost and GPUs required by transformer-based model. To mitigate this risk, only tuned dropout size and batch size are used in this research work.

9.2.2 External validity

One potential threat to external validity is the dataset used in our research work. The experimental dataset is created by considering approach in mind. From Tables 9 and 10, the cross-validation techniques (shown in the last two rows of both tables) indicate generalizability of the proposed approach. Other threat to external validity is this research work classify reviews from specific user feedback platform such as Apple store and Google Play store, and cannot classify reviews from other platform such as Amazon store etc. also.

This research paper empirically examines the ability of pre-trained language model to classify requirements from app reviews. App reviews classification mainly focuses on extracting useful information in the context of software engineering, which can be used by app developers for requirements engineering, release planning and for software maintenance. Our study is therefore limited to app review classification for software engineers and this work does not study pre-trained language model for other purposes, such as finding issues in business perspective, product reviews, and application such as attention mining.

9.2.3 Conclusion validity

A threat to statistical conclusion validity might occur due to the use of imbalanced datasets. In this case, multi-label datasets have different classes. Unfortunately, undersampling and oversampling techniques are only available for binary classification. For multi-label multi-class, these techniques still needs more exploration.

9.2.4 Construct validity

A threat to construct validity is that the labels in the exploited dataset may be incorrect. Jha and Mahmoud [25] manually labeled the dataset with the help of human annotators,the labeled dataset could be incorrect for different reasons. Consequently, such incorrect labeling can produce inaccurate results.

10 Conclusion

Nowadays, online user feedback consists of valuable information that can be used for next release planning. Automatic analysis of these feedback helped developers to understand the user's perspective. Existing keyword-based method is not suitable for multi-label review classification and fails to classify user reviews with reasonable performances (i.e., they do not achieve high recall with acceptable precision, which is necessary for RE). Moreover, we did not find any paper that applied deep learning to multi-label dataset of user reviews. To overcome this research gap, this research work also implemented deep learning techniques. Then, we introduced MNoR-BERT framework to improve the state-of-the-art results. The proposed approach is used to predict important user concerns about their apps using transfer learning-based models. This paper addresses the research gaps and compares MNoR-BERT with machine and deep learning techniques. Extensive experiments are conducted on dataset of 6000 user reviews to evaluate the performance of proposed approach. The results obtained by the model are compared with those of four baseline adaptation algorithms (BR_{NB}, BR_{SVM}, BR_{KNN}, and BR_{DT}) and neural networks (CNN, LSTM, and BiLSTM) with GloVe representation. The reported results indicate that MNoR-BERT performs better than state-of-the-art multi-label classification methods. Moreover, we compared our model with binary relevance for machine learning techniques (**RQ1**), dictionary/keyword-based approach (**RQ2**), and with two traditional word representation techniques, that is, TF-IDF, and GloVe (**RQ3**). From these research questions, we conclude that MNoR-BERT outperforms all approaches. In addition, BERT model can be easily trained with less number of epochs, whereas deep learning models need more epochs means require more data to learn. We also investigate that the performance of the model is affected when the number of classes increases. We also found that existing techniques need pre-processing

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

for user reviews to convert them into specific model format; however, our proposed model can be easily accessed by developers without applying any preprocessing technique. Our Research work helps to extract the important NFRs that are important for developers, stakeholders and analysts.

We envisage future research work in multiple directions, first is to extend neural language models for classification improvements. Second, is to investigate effectiveness of other pre-trained language models on multi-label review classification problem. Moreover, ensemble methods can also be used for enhancing the classification accuracy.

Data Availability The data of this study are public and available on link [http://seel.cse.lsu.edu/data/emse19.zip].

Declarations

Conflict of interest The authors declared that they have no conflict of interest.

References

1. Achimugu P, Selamat A, Ibrahim R, Mahrin MN (2014) A systematic literature review of software requirements prioritization research. Inform Softw Technol. https://doi.org/10.1016/j.infsof.2014.02.001
2. Araujo A, Golo M, Viana B, Sanches F, Romero R, Marcacini R (2020) From bag-of-words to pre-trained neural language models: improving automatic classification of app reviews for requirements engineering. Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional. https://doi.org/10.5753/eniac.2020.12144
3. Aslam N, Ramay WY, Xia K, Sarwar N (2020) Convolutional neural network based classification of app reviews. IEEE Access 8:185619–185628. https://doi.org/10.1109/ACCESS.2020.3029634
4. C Baker, L Deng, S Chakraborty, J Dehlinger, (2019) Automatic multi-class non-functional software requirements classification using neural networks. In: Proceedings–Int Comput Softw Appl Conference, 2: 610–615, https://doi.org/10.1109/COMPSAC.2019.10275
5. Benites F, Sapozhnikova E (2016) HARAM: a Hierarchical ARAM neural network for large-scale text classification. In: Proceedings–15th IEEE International Conference on Data Mining Workshop, ICDMW 7: 847–854. https://doi.org/10.1109/ICDMW.2015.14
6. Binkhonain M, Zhao L (2019) A review of machine learning algorithms for identification and classification of non functional requirements. Expert Syst Appl. https://doi.org/10.1016/j.eswax.2019.100001
7. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Transact Assoc Comput Linguist 5:135–146. https://doi.org/10.1162/tacl_a_00051
8. Boutell MR, Luo J, Shen X, Brown CM (2004) Learning multi-label scene classification. Pattern Recognit. https://doi.org/10.1016/j.patcog.2004.03.009
9. Boehm B, In H (1996) Identifying quality-requirement conflicts. IEEE Softw 13:25–35
10. Chen N, Lin J, Hoi SC, Xiao X, Zhang B (2014) AR-miner: mining informative reviews for developers from mobile app marketplace. In: Proceedings of the 36th international conference on software engineering. https://doi.org/10.1145/2568225.2568263
11. Chollet F (2021) Deep learning with python. Simon Schuster
12. Chung L, Prado Leite JC (2009) On non-functional requirements in software engineering. Springer, Berlin, Heidelberg, Conceptual modeling Foundations and applications. https://doi.org/10.1007/978-3-642-02463-4_19
13. Ciurumelea A, Schaufelbühl A, Panichella S, Gall HC (2017) Analyzing reviews and code of mobile apps for better release planning. In: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). https://doi.org/10.1109/SANER.2017.7884612
14. Clare A, King RD (2001) Knowledge discovery in multi-label phenotype data. European conference on principles of data mining and knowledge discovery. Springer, Berlin, Heidelberg, pp 42–53
15. Cleland-Huang J (2007) Quality requirements and their role in successful products. In: 15th IEEE International Requirements Engineering Conference. https://doi.org/10.1109/RE.2007.45
16. Dai Z, Yang Z, Yang Y, Carbonell J, Le QV, Salakhutdinov R (2019) Transformer-xl: attentive language models beyond a fixed-length context. ACL 2019–57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference.https://doi.org/10.18653/v1/p19-1285
17. de Araújo AF, Marcacini RM (2021) RE-BERT: automatic extraction of software requirements from app reviews using BERT language model. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing.
18. Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, 1(Mlm).
19. Elisseeff A, Weston J (2001) A kernel method for multi-labelled classification. In: Advances in neural information processing systems
20. Genc-Nayebi N, Abran A (2017) A systematic literature review: opinion mining studies from mobile app store user reviews. J Syst Softw. https://doi.org/10.1016/j.jss.2016.11.027
21. Gibaja E, Ventura S (2014) Multi-label learning: a review of the state of the art and ongoing research. Data Mining and Knowledge Discovery, Wiley Interdisciplinary Reviews. https://doi.org/10.1002/widm.1139
22. Gnanasekaran RK, Chakraborty S, Dehlinger J, Deng L (2021) Using Recurrent Neural Networks for Classification of Natural Language-based Non-functional Requirements. REFSQ Workshops
23. Groen EC, Kopczyńska S, Hauer MP, Krafft TD, Doerr J (2017) Users—the hidden software product quality experts?: A study on how app users report quality aspects in online reviews. In: 2017 IEEE 25th international requirements engineering conference (RE). IEEE pp 80–89. https://doi.org/10.1109/RE.2017.73
24. Guzman E, Maalej W (2014) How do users like this feature? a fine grained sentiment analysis of app reviews. In: 2014 IEEE 22nd international requirements engineering conference (RE), https://doi.org/10.1109/RE.2014.6912257
25. Jha N, Mahmoud A (2019) Mining non-functional requirements from app store reviews. Empir Softw Eng 24(6):3659–3695. https://doi.org/10.1007/s10664-019-09716-7
26. Kurtanović Z, Maalej W (2017) Automatically classifying functional and non-functional requirements using supervised machine

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

learning. In: 2017 IEEE 25th International Requirements Engineering Conference (RE) IEEE. https://doi.org/10.1109/RE.2017.82
27. Kurtanović Z, Maalej W (2018) On user rationale in software engineering. Requirements Eng. https://doi.org/10.1007/s00766-018-0293-2
28. Liu SM, Chen JH (2015) A multi-label classification based approach for sentiment classification. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2014.08.036
29. Lu M, Liang P (2017) Automatic classification of non-functional requirements from augmented app user reviews. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, https://doi.org/10.1145/3084226.3084241
30. Maalej W, Kurtanović Z, Nabil H, Stanik C (2016) On the automatic classification of app reviews. Require Eng. https://doi.org/10.1007/s00766-016-0251-9
31. Maalej W, Nabil H (2015) Bug report, feature request, or simply praise? on automatically classifying app reviews. In: IEEE 23rd international requirements engineering conference (RE). https://doi.org/10.1109/RE.2015.7320414
32. Madjarov G, Kocev D, Gjorgjevikj D, Džeroski S (2012) An extensive experimental comparison of methods for multi-label learning. In: Pattern recognition. https://doi.org/10.1016/j.patcog.2012.03.004
33. McIlroy S, Ali N, Khalid H, E Hassan A (2016) Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. In: Empir Softw Eng. https://doi.org/10.1007/s10664-015-9375-7
34. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: 1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings.
35. Navarro-Almanza R, Juarez-Ramirez R, Licea G (2017) Towards supporting software engineering using deep learning: A case of software requirements classification. In: 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT) pp. 116–120 .https://doi.org/10.1109/CONISOFT.2017.00021
36. Pagano D, Maalej W, 2013 User feedback in the appstore: an empirical study. In: 2013 21st IEEE international requirements engineering conference (RE). IEEE
37. Palomba F, Linares-Vásquez M, Bavota G, Oliveto R, Di Penta M, Poshyvanyk D, De Lucia A (2018) Crowdsourcing user reviews to support the evolution of mobile apps. J Syst Softw. https://doi.org/10.1016/j.jss.2017.11.043
38. Panichella S, Di Sorbo A, Guzman E, Visaggio CA, Canfora G, Gall HC (2015) How can i improve my app? Classifying user reviews for software maintenance and evolution. In: 2015 IEEE international conference on software maintenance and evolution (ICSME) . https://doi.org/10.1109/ICSM.2015.7332474
39. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J (2011) Scikit-learn: Machine learning in Python. J Mach Learn Res 12:2825–2830
40. Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).
41. Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding by generative pre-training.
42. Stanik C, Haering M, Maalej W (2019) Classifying multilingual user feedback using traditional machine learning and deep learning. In: Proceedings–2019 IEEE 27th International Requirements Engineering Conference Workshops. https://doi.org/10.1109/REW.2019.00046
43. Sechidis K, Tsoumakas G, Vlahavas I (2011) On the stratification of multi-label data. Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Berlin, Heidelberg, pp 145–158
44. Di Sorbo A, Panichella S, Alexandru CV, Visaggio CA, Canfora G (2017) SURF: summarizer of user reviews feedback. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), IEEE. https://doi.org/10.1109/ICSE-C.2017.5
45. Tsoumakas G, Katakis I, Vlahavas I (2010) Random k-labelsets for multilabel classification. IEEE Transact Knowl Data Eng. https://doi.org/10.1109/TKDE.2010.164
46. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inform Process Syst 30:5999–6009
47. Winkler J, Vogelsang A (2017) Automatic classification of requirements based on Convolutional neural networks. In: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW) IEEE. https://doi.org/10.1109/REW.2016.16
48. Wu J, Gao Z, Hu C (2009) An empirical study on several classification algorithms and their improvements. In: International Symposium on Intelligence Computation and Applications. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04843-2_30
49. Yang H, Liang P (2015) Identification and Classification of Requirements from App User Reviews. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE. https://doi.org/10.18293/SEKE2015-063

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com