BERT-RCNN: an Automatic Classification of App Reviews using Transfer Learning based RCNN Deep Model

Kamaljit Kaur (Kamaljitcs.rsh@gndu.ac.in)
Guru Nanak Dev University
Parminder Kaur
Guru Nanak Dev University

Research Article

Keywords: Software engineering, Requirements Engineering, App review classification, deep learning, Transfer learning

Posted Date: January 25th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2503700/v1

Additional Declarations: No competing interests reported.

BERT-RCNN: an Automatic Classification of App Reviews using Transfer Learning based RCNN Deep Model

Kamaljit Kaur*, Parminder Kaur
Department of Computer Science, Guru Nanak Dev University, Amritsar, India
Kamaljitcs.rsh@gndu.ac.in, Parminder.dcse@gndu.ac.in

Abstract

App reviews considered as the major concern over the internet. It influences the user's mind before purchasing the app. However, such user reviews might contain technical information about the app that can be valuable for the developers and software companies. Due to pervasive use of mobile apps, large amount of data is created by users on daily basis. Manual identification and classification of such reviews is very time-consuming and laborious task. Hence, automating this process is essential for helping developers in managing these reviews efficiently. Recently, traditional machine learning and deep learning oriented approaches have been exploited for classification of app review into software requirements. Most of the machine learning techniques utilizes traditional word techniques to extract the features from the textual reviews. In addition, deep learning techniques with efficient word embedding technique also improve the classification performance of the model. However, it is found that these techniques suffer from polysemic problem. To address aforementioned problem, this research paper presents novel neural language based framework, BERT-RCNN to classify app reviews into specified categories of functional requirements such as Bug report, Feature Request and Relevant etc. The presented framework has two main modules: Bidirectional Encoder Representation for Transformer (BERT) module and Recurrent-Convolutional Neural Network (RCNN) module. BERT module extracts the contextual relationship between the textual reviews. Then, these features are input to RCNN module to capture more important features with deep semantic information. The output of RCNN is fed to fully connected layer for classification purpose. Extensive experiments are conducted on five datasets to investigate the effectiveness of the presented BERT-RCNN model. For evaluation purpose, standard performance measures are used to obtain the results. Moreover, hyperparameter such as BiLSTM memory unit, CNN filters and CNN kernel size are tuned to ensure the quality of prediction. Comparative analysis of proposed model is conducted with existing state-of-the-art models. It shows that BERT-RCNN outperforms other state-of-the-art models with respect to precision, recall, f-measure and accuracy.

Keywords: Software engineering, Requirements Engineering, App review classification, deep learning, Transfer learning

1. Introduction

With the rapid growth of app stores such as App store, Google play, Blackberry world stone, Microsoft phone app store, users have become protagonists of internet, as they easily express information or views on these platforms. This information contain rich source of information that have had tremendous effect on software engineering practices. For example, (i) user reviews influences the initial thoughts of other users, (ii) developers can introduce new feature and improve their app by fixing bugs for next release planning. In addition, issues in app gradually lose its popularity and leads to failure of popular app. Therefore, identifying and understanding user's needs considered as the major challenge for the developers. Online platform receives millions of reviews on daily basis. Manual extraction of relevant reviews is impractical and labor-intensive task. Most of the earlier studies have developed rule-based, machine learning and deep learning techniques for automatic extraction of useful reviews from user feedback. Prior studies mainly exploit classification methods for detection relevant reviews such as app problems, feature requests, and aspect evaluation etc.

Most of the earlier methods for app review classification were based on rule-based and machine based methods. The dictionary based approach is used to extract important features. Besides, machine-learning based methods are utilized by various researchers, but these supervised classification methods need labelled data for training. Therefore, various research works rely on shallow models designed in such a way to obtained satisfactory results. Such models utilize text representation techniques such as Bag-of-word, TF-IDF and word2vec techniques. However, these techniques suffer from polysemic problem, which means it ignores the syntactic structure of the sentence. In addition, these models suffer from poor generalization problem.

To overcome aforementioned issues, neural language models surprisingly surpassed the predictive performance of traditional machine and deep learning models. These models reported the state-of-the art for various Natural Language Processing (NLP) tasks, and reported promising predictive results on various NLP tasks such as question answering, text classification etc. Inspired by the performance of neural language Bidirectional Encoder Representations from Transformers (BERT) in Hey et al., (2020), this research work proposes the BERT-RCNN model for app review classification. In this framework, most popular BERT pre-trained language model is used to calculate the corresponding contextual representation. Moreover, BERT model is fine-tuned by adding deep layers to boost the performance of app review classification by extracting the important features from contextual representation. The proposed framework consist of three parts: one is BERT layer as word embedding layer for contextual representation of app reviews, other module contains deep layers such as RNN and CNN layers, here variant of RNN i.e BiLSTM extracts important features and CNN selects the feature with maximum value. Finally, softmax classifier is employed to classify app reviews. The input of the model is textual app review and output is the predictive label of the app review. The predictive performance of the proposed model has been employed on five app review classification benchmark dataset. To prove the validation of proposed model, this research work compared proposed model with existing studies and other NLP related deep neural architecture for app review classification. Experimental analysis demonstrates that the presented BERT-RCNN model can outperform the state-of-the-art results for classification task.

The major contributions of this research work are as follows:

- This research work proposes a novel transfer learning based deep learning architecture, which combines Bidirectional LSTM and Convolutional Neural Network in a way to improve the performance of classification model. Two BiLSTM hidden layers are connecting in opposite directions to same context. The presented Bidirectional convolutional layers are stacked over contextual information capture by BERT model. The deep learning layers are used to improve the learning of sub-features while simultaneously reducing their feature space.
- This paper employs an efficient text representation technique containing the entire contextual information of word, whereas the existing word representation techniques suffer from polysemic problem.
- This paper employs BiLSTM layers to capture the entire contextual information by combining local and global features.
- Prior supervised model related studies trained model on labeled dataset. However these models start decreasing performance as soon as applied on other dataset. To overcome this problem, BERT model is utilized to extract the contextual relationship between the words.
- To best of our knowledge, this is the first study in which BERT model is improved by the RCNN layers for app review classification. The empirical results indicate that the utilization of these layers can outperform the attention mechanism based model used in deep learning architectures.
- The predictive performance of proposed model scheme has been employed on five app review classification benchmark datasets.

The paper is organized as follows: section 2 discusses the related work and research motivation for this research work. Section 3 and 4 presents the preliminaries and architecture of the proposed model respectively. Experimental evaluation and results are discussed in section 5. Finally, we conclude the paper in section 6 along with future directions in section 7.

2. Related work

Most traditional app review categorization research works used topic modeling approach as core classification method(Campbell et al., 2015). Initially, topic modeling is discussed by Chen et al., (2014) for summarization of informative reviews. Nayebi et al., (2018) have utilized topic modeling to analyze the correlation between tweets and app reviews. In addition, their approach aims to remove uninformative reviews. Gao et al., (2018) have presents topic modeling approach that automatically identifying the emerging app issues. Hadi and Fard, (2020) also presented topic modeling based Adaptive Online Biterm Topic Model (AOBTM) approach to analyze short texts. In addition, Topic Modeling approaches attracts the attention of requirements engineer in different fields such as categorization, identification, prioritization and sentiment analysis of app reviews in recent years (Gao et al., 2018, Hadi and Fard, 2020, Guzman and Maalej, 2014,Yang et al., 2021, Wardhana and Sibaroni, 2021).

Over the last decade, few researchers have done lot of work in automatic classification of app reviews. Chen et al., (2014) proposed a review analytics tool AR-Miner for mining and summarizing the informative reviews. AR-Miner tool make use of Naïve Bayes algorithm for mining relevant reviews by classifying them as “informative” or “not relevant”. Then, topic modelling is used to summarize the reviews into different groups with similar content. Finally, Review ranking scheme is applied to rank these groups. The authors have validated their approach on four android apps and achieve good results in precision, recall and F1-measure. But they have not generalized their approach on other set of apps or app stores. Fu et al., (2013) have designed Wiscom tool for automatic identification of incorrect rated reviews. They discussed the ten topmost factors that affect the success of the mobile application. Then, Oh et al., (2013) developed Support Vector Machine (SVM) based review digest system, which automatically classify user reviews into functional and non-functional requirements. Gu and Kim,(2016) have proposed SUR-Miner approach, to summarize and classify reviews into five categories. Shah et al., (2019) have compared simple Bow with complex feature extraction (CNN) based model. They evaluated their approach on reviews of 17 different apps and conclude that simple BOW performs better than expensive CNN model. Rustam et al., (2020) have developed model in conjunction with machine learning techniques. They reported that combination of features improve the performance of classifier. Maalej and Nabil, (2015) have also extract different features with machine learning techniques, and classify user reviews into four classes: bug report, feature request, user experience, and rating.

McIlroy et al., (2016) have presented an automated multi-labeling scheme to assign multilabel to each review that can help app developers. The authors have analyzed 601,221 reviews from 12,000 apps in Google play store and discovered that 30% of reviews contain various types of user complaints (such as feature requests, functional complaints, and privacy issues). Then, they have applied their automated multi-label approach with corresponding user complaints. Maalej et al., (2016) have investigated machine learning techniques for automatically mining of user rationale. The authors have discussed the grounded theory approach for identification and justification of user issues such as upgrading, installing, or switching in software application. They have validated their approach on labeled dataset of 32,414 user reviews of Amazon store. They concluded that user frequently raised issues related to compatibility, performance, and usability. The authors have evaluated their approach by utilizing machine learning techniques such as SVM, NB, and Logistic Regression. The reported results showed that different rationale concepts can be detected with high level of precision of 80% and recall of 99%. Villarroel et al., (2016) have introduced **C**rowd **L**istener for rele**A**se **P**lanning (CLAP) for automatic clustering of user reviews, and then prioritizing the clusters for the future release planning. The authors have discussed only three issues such as (i) bug report, (ii) feature request and (iii) others. Later, Scalabrino et al., (2019) have improved a web application based CLAP tool for automatic extraction of user reviews and cluster the reviews that reported same issues. Their proposed framework utilize random forest algorithm to categorize the reviews into seven types: bug report, feature request, performance, energy, security, usability and others. Then, density based algorithm DBSCAN is used to cluster the related reviews. Finally, classifier is used to assign high or low priority to cluster based on the size, and average length of the review. The cluster with higher priority can be recommended to app developers for the next release. The authors have evaluated their approach in industrial settings and concluded that CLAP achieves an average of 75% of clustering in bug report and 83% in feature request.

2.2 Deep models for app review classification

In this field of app review classification, most research studies have been oriented towards deep neural network based models. Stanik et al., (2019) have used deep learning approaches to classify multilingual user feedback (English and Italian). The authors have performed comparative analysis between traditional machine learning and deep learning approaches. The reported results demonstrate that machine learning still achieve comparable results than deep learning techniques. Aslam et al., (2020) have proposed convolutional neural network deep learning model for app review classification. Their approach extracts textual and non-textual reviews from public dataset. The authors conclude that their approach performs significantly better than state-of-the-art techniques in terms of precision, recall and f1-measure. Traditional machine and deep learning techniques suffers from poor generalization i.e. the performance of the model decreases on unseen data. To alleviate aforementioned problem, Qiao et al., (2020) have proposed domain-oriented deep learning approach that extracts more critical reviews (such as new feature, bug report etc.) from online user reviews. In their model, they incorporated embedding layer to extract syntactic features and then CNN and RNN layer are added to identify relevant details from the reviews. The authors have conducted comparative analysis of machine and deep learning techniques in classification of app reviews. The reported results demonstrate that deep learning methods perform better than machine learning. Henao et al., (2021) have applied BERT model for classification of app reviews into problem report, feature request, and irrelevant. In their paper, they improved classical machine learning and deep learning model using pre-trained language model. The authors have also performed comparative analysis and conclude that BERT achieves highest performance in precision and recall than other baseline methods. Hadi and Fard, (2020) have transfer learning based visualization tool for automatic identification of reviews related to energy efficiency of mobile apps. Topic modeling technique is used to extract the main topics discussed by the reviews. He et al., (2019) have proposed LSTM based model for spam detection in app reviews. Haering et al., (2021) have introduced deep learning based DeepMatcher approach to extract problem reports from app reviews and matching these reports with the bug reports identified by the development team. Their approach helps developers with early identification of bugs. Their approach eventually supports developers to update their products by enhancing the way to predict the duplicate or similar reviews.

2.3 Research Design

2.3.1 Motivation

The motivation towards the research work may be described as follows:

In the digitalization era, mobile phone applications on the web are considered as communication channel between users and developers. A huge amount of user textual reviews are received on these platforms contain valuable information about the app issues such as bug report, new feature request and ratings etc. These reviews have implicit knowledge of users' expectation and complaints and such information is essential for software developers in decision-making process. This knowledge can be utilized in software development life cycle i.e. from requirements engineering to software maintenance by software companies. The role of app reviews in software engineering is discussed by Martin et al. and Sarro et al. From the prior reports, 70% of the reviews contain irrelevant and noisy reviews. Therefore, early identification of these noisy reviews can improve the next-release version of app. These reviews can be seen from two perspectives, first, such user reviews influences their initial thoughts before downloading the app. Positive reviews attract more users and hence bring the financial gains.

Another major challenge is ambiguity of words in app review. It contains proverb and short text reviews, where actual user concern or issue is quite difficult to predict. Traditional feature engineering techniques require proper preprocessing and domain-knowledge. Prior studies claimed that deep neural language model can reduce the burden of feature engineering. Therefore, this paper adopts the BERT language model to extract the contextual information, and LSTM is adopted to store the temporal contextual information by retaining the long-term dependencies of words inside the review sentence. CNN is adopted for the extraction of local features and patterns. It is mainly suitable for better classification task.

Traditional machine learning techniques can be used in feature engineering, but these techniques are not suitable for large volume of data. However, neural language model can better extract the features but not suitable to handle long-sequential information, which is essential for app review classification. In the second phase, CNN layer can be utilized to extract feature with maximum value, but if initial BERT layer is unable to capture the contextual information, then CNN layers also fails to extract the important features from the review sentence. This has motivated us to use LSTM layer in first place and CNN is on the second place.

Existing studies still used topic and rule-based methods to classify app reviews into software requirements. Due to informal texts and grammatical errors in reviews, new rules and topics needs to be created, that leads to increase in cost and also conflict can be occur between rules in few cases. This is challenging for the the developers to maintain. Although, few researchers reported that traditional machine learning with feature extraction techniques (TF-IDF, word2vec, GloVe etc.) achieve remarkable results. But, these models suffer from the main drawbacks: (i) feature space generated by these techniques has high dimensionality (ii) these models require good amount of data for training. (iii) These techniques did not retain the syntactic structure of sentence. (iv) Such models require manual feature extraction, therefore, feature engineering is considered as labor-intensive and time consuming process.(v) these techniques suffers from out-of-vocabulary problem. To alleviate these issues, BERT model is utilized to extract the contextual relationship between the words of sentence.

These pre-trained language models are exploited in natural language processing. Word2vec and GloVe word representation techniques are also achieve promising proximity between words, but without retaining the word orders. This is also current problem of existing models. Traditional machine learning techniques cannot trained well on imbalanced datasets, and eventually susceptible to errors. This is the reason why we have used deep learning models. Few researchers have utilized pre-trained language model for app issue classification. Hadi and Fard, (2020) reported a lite BERT model achieves satisfactory results than other pre-trained models, whereas Henao et al., (2021) have claimed that simple BERT model does not leads to performance gain. To fill these gaps, our research work fine-tuned BERT model by integrating with deep learning layers.

3. Preliminaries

In this section, overview of basic building blocks of proposed model is discussed.

3.1 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representation from Transformer (BERT) is type of transfer learning designed by Google (Devlin et al., 2019). BERT, also known as neural language model generate output based on the context of the input. In this paper, BERT model utilized as word embedding layer to convert the text into vectors. The main purpose of BERT is to learn text from left to right by jointly configuring the context of text. Traditional encoder-decoder with attention mechanism models suffers from trivial prediction problem i.e during training process, target word sees itself. Therefore, BERT model is trained with two different tasks such as Masked Language Model (MLM) and Next Sentence Prediction (NSP). In MLM, some input tokens are randomly replaced by [MASK] token, then BERT model was trained to predict the masked token on the basis of the context. For example, “love this camera app”, input tokens can be randomly replaced as, “love this [MASK] app”. The NSP technique exhibits the relationship between two sentences for example in question answering. In addition, model predict sentence B, that is likely to be adjacent to sentence A. In conclusion, the prominent goal of these tasks is to predict the word based on its contextual meaning. The graphical representation is shown in fig 1.

Since the BERT model is employed on raw input. Each word must be embedded into tokens. Therefore, input embedding of BERT model itself consists of three embedding: token embedding, position embedding and segment embedding. The raw input sequence is enclosed with special tokens such as [CLS] and [SEP] tokens indicate the beginning and end of input sequence. In this paper, BERT base model (L=12, H=768, A=12, Total parameter=110M) is utilized to train the model. Here, L is the transformer blocks, H is the hidden layer and A is the self-attention mechanisms. The output of the BERT model is sequence of vector containing contextual information.

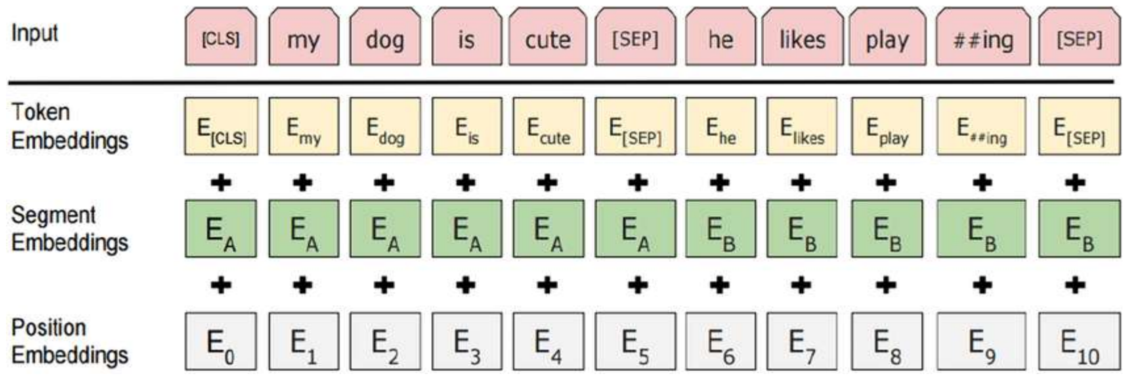

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Fig 1: The visual representation of BERT Layer in proposed model. Source: adopted from (Devlin et al., 2019).

3.2 Long-Short term memory

LSTM is a variant of Recurrent Neural Network (RNN) designed to obliterate the exploding or vanishing gradient problem. LSTM unit consists of memory cell and gates, namely input gate i_t, forget gate f_t and output gate o_t shown in fig 2. These gates are used to store and regulate the information from the cell at specific time-interval. The vector representation of LSTM has been carried out from the following equations.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3}$$

$$v_t = tanh(W_v x_t + U_v h_{t-1} + b_v) \tag{4}$$

$$c_t = f_t . c_{t-1} + i_t . v_t \tag{5}$$

$$h_t = o_t . \tanh(c_t) \tag{6}$$

Where x_t denotes the input to the LSTM unit,i_t f_t, and o_t are the activation vector of input, forget and output gates respectively. Also, c_t and v_t represent the cell state and hidden state vector respectively. σ , Tanh and (.) are the sigmoid, tangent and product functions. In addition, W indicate the weight associated with the corresponding vectors and b is depicted as bias vector. After obtaining the future context along with the preceding context, BiLSTM layer combines forward and backward hidden layers. Due to the flow of temporal information in both directions improve the learning ability of the network

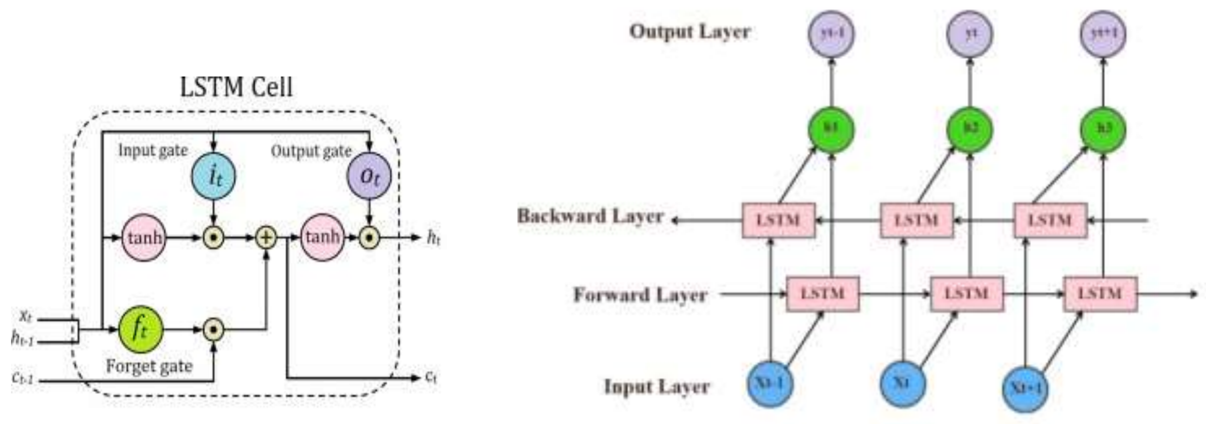


Fig 2: The graphical representation of (a) LSTM and (b) BiLSTM cell. Source adopted (a) from (Basiri., 2021)

3.3 Convolutional Neural Network

Convolutional Neural Network consists of series of convolutional layers to extract local features from NLP applications. The convolution filter is slide on vector space to calculate the dot product between the filter and word vector. Then, max-pooling layer is used to capture the important feature with maximum value. These features are fed to fully connected layer for classification purpose.

4. Proposed Model

In this section, we propose a novel deep neural network based framework called RCNN-TL for app review classification. Fig 3 presents the outline of the proposed model. The model consists of five parts: data processing, embedding layer, Bidirectional layer, convolutional layer, pooling layer and fully connected layer. Initially, neural language model such as BERT used as embedding layer to obtain contextual information from app reviews. Then, bidirectional and convolutional layers have been employed to capture important features from contextual information. Pooling layer is stacked over deep layers to reduce the dimensionality of feature space. Finally, softmax layer has been employed for the prediction of classes on the basis of probabilities. The detail of each part of architecture has been discussed as follows.

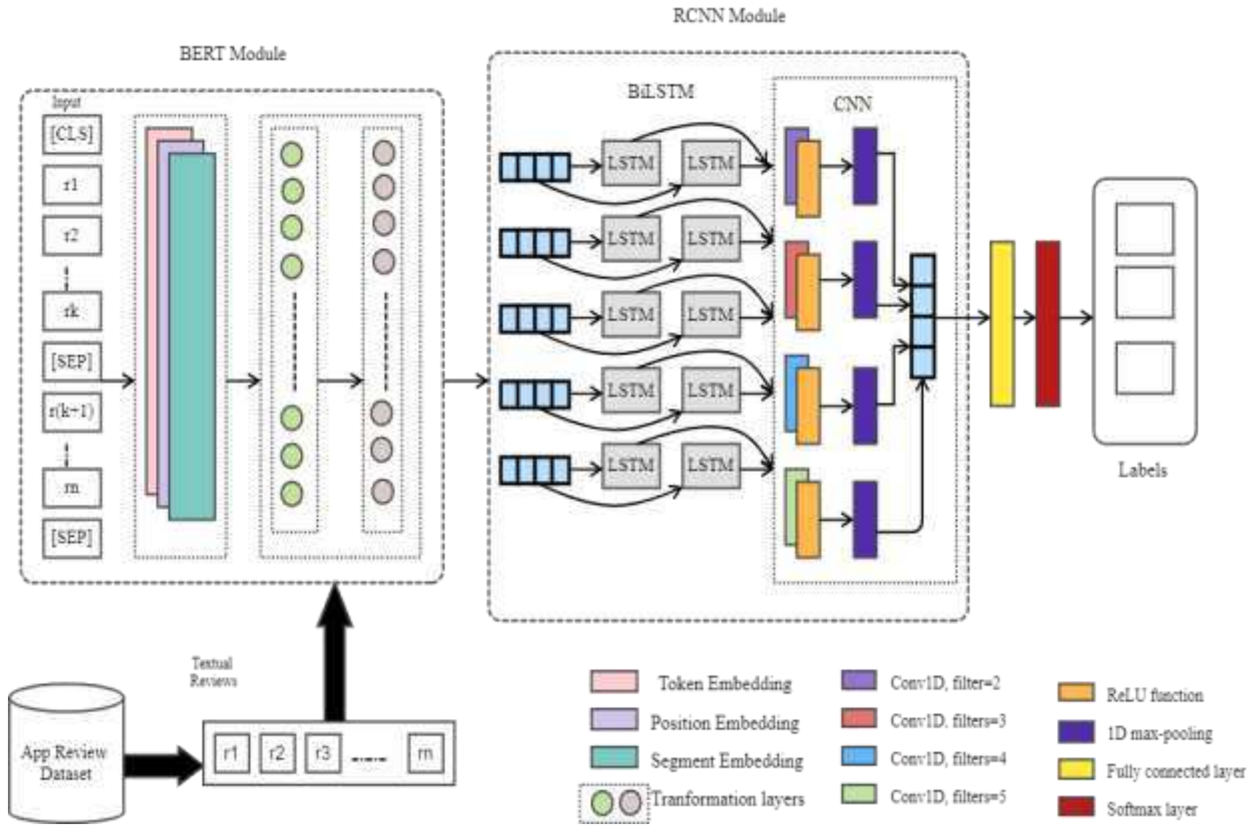


Fig 3: BERT-RCNN framework

4.1 The embedding layer

In order to obtain deeper contextual representation, we utilized BERT model to generate contextual feature vector representation from textual reviews. For an n dimensional textual reviews, special token such as start token [CLS] and separation token [SEP] are inserted. Then, token, segment, and position embeddings are summed up and fed into complex deep model of transformers to obtain final context feature vector representation depicted as E= [E_1, E_2 ,.., E_N, E_{N+1}].

4.2 Bidirectional LSTM layer

In existing literature, Recurrent Neural Network (RNN) shows promising results in sequential modelling (Natural language processing). By following the same vein, this paper has employed LSTM to extract more high-quality features in both directions one is forward and other is backward. The forward and backward LSTM receives output of previous layer and process from left to right and right to left, respectively (eqn 7 and 8). Then, these two modules are concatenated to achieve final output (eqn 9). We employ LSTM to improve BERT layer by containing previous and future semantics information with deeper understanding.

$$\vec{L}_{LSTM} = \overrightarrow{LSTM}\,(w_i E_i), i \in [0, t] \tag{7}$$

$$\overleftarrow{L}_{LSTM} = \overleftarrow{LSTM}\,(w_i E_i), i \in [t, 0] \tag{8}$$

$$L_{LSTM} = [\vec{L}_{LSTM}, \overleftarrow{L}_{LSTM}] \tag{9}$$

4.3 The Convolutional layer

The convolutional layer is employed on obtained context feature matrix $L \in R^{l \times m}$ in Bi-LSTM mechanism, where l is the number of words, and m is the embedding dimensions. The main purpose of CNN layer is to extract local features by performing convolutional operation on the input L. This convolutional operation is repeatedly applied on matrix to generate feature map. For each feature map, non-linear RELU function (f) is applied on L to accurately extract local context. From eqn (10), dot (.) operator indicates the convolutional operation and b depicted as bias. In addition, sub-feature matrices such as $\left(l_{1:p}, l_{2:p}, \ldots l_{(l-p+1):l}\right)$ are created via sliding convolutional window $C \in R^{p \times m}$ (where p is the size of the kernel). Then, the visualized description of final feature map representation is shown in eqn (11)

$$f_i = f(C.L_{i:(i+p-1)} + b) \tag{10}$$

$$F = [f_1, f_2, \ldots f_{(l-p+1)}] \tag{11}$$

4.4 The pooling layer

In this step, maximum-pooling layer is stacked over feature maps generated by CNN layer. These feature maps contain more deep semantic information by eliminating the ambiguity of the word. The main objective of this layer is to extract the feature map with the largest value (eqn 12). In this research paper, only one max-pooling layer is employed. The outputs of N extracted features are concatenated to feature vectors (eqn 13) and fed to fully connected layer.

$$\bar{F} = \max\{F\} \tag{12}$$

$$\bar{\bar{F}} = [\overline{F_1}, \overline{F_2}, \ldots, \overline{F_N}] \tag{13}$$

4.5 Feed forward layer

In order to obtained final representation of feature vectors, feed forward dense layer are used with *RELU* activation function. Due to multi-class nature of dataset, softmax activation function is utilized at the classification head of the model. The main objective of this layer is to reduce the dimensions to the number of classes (q). In addition, categorical cross-entropy loss function with Adam optimizer is used to prevent the overfitting. The summarization of proposed algorithm gives in Algorithm 1.

Algorithm 1: Pseudo code of proposed framework (BERT-RCNN)

```
1. For each n dimensional app review, BERT as word embedding transforms the words into contextual word embedding matrix.
2. Employ one bidirectional-LSTM layer to obtain both previous and future contextual features $\overleftarrow{L}_{LSTM}, \overrightarrow{L}_{LSTM}$ respectively using eqns (7-9).
3. Employ convolutional neural network on output of Bi-LSTM to capture the important features using eqn (10) and (11)
4. Employ one global Maximum pooling layer to select the feature with maximum value from local contextual feature matrix using eqn (12) and eqn (13).
5. Combined all feature vectors selected by pooling layer and feed to fully connected layer
6. Add two dense layer with RELU activation function
7. To get final contextual representation, feature vectors are feed into softmax classifier
8. Update the parameters of the model using categorical cross-entropy loss function with Adam optimizer.
```

5. Experimental evaluation and results

In this section, proposed approach evaluated on available dataset of app reviews followed by baseline methods and evaluation measures of proposed model. In addition, extensive experiments are conducted to evaluate the performance of the model.

5.1 Datasets

Dataset 1: This dataset consists of 34,000 app reviews of 17 Google play apps from 16 different categories. The authors have manually annotated the reviews and classified reviews into five categories such as: aspect evaluation, praise, feature request, bug report, and others (Gu and Kim, 2016).

Dataset 2: The dataset contains app reviews from Google play and tweets. This dataset is also manually labeled by human annotators and categorized the app reviews into three classes, namely, problem report, inquiry and irrelevant Stanik et al., (2019).

Dataset 3: The dataset consists of 1,259 reviews collected from Google play and App store. The authors have discussed four categories of app reviews i.e. usability, reliability, portability, performance (Lu and Liang, 2017).

Dataset 4: This dataset is created by Maalej et al., (2016). It contains 4,400 app reviews Google play and App store from top apps in different categories. The app reviews are labeled with four categories: Bug report, feature request, user experience and rating.

Dataset 5: It consists of 1,500 app reviews obtained from Google play and app store (Guo and Singh, 2020)

Table 1 illustrates the brief description of datasets used in this research work for the performance analysis of proposed model and state-of-the-art models.

Table 1: Distribution of the classes in datasets along with the average length of reviews, vocabulary size and test size

Datasets	Total number of reviews	Number of classes	Average length of reviews	Vocabulary size	Test size
Dataset 1	34,000	5	8.095	8,519	8,500
Dataset 2	6,391	3	23.889	6,499	2,237
Dataset 3	1,259	4	13.4678	1,566	441
Dataset 4	4,400	4	101.925	4,215	922
Dataset 5	1,500	3	6.015	1,223	375

5.2 Experimental setup

In this section, different parameters and experimental setup of this research work is discussed.

The proposed model was designed using Keras with Tensorflow as backend, Scikit-learn, Natural Language Toolkit (NLTK) python in-build libraries (Chollet et al., 2015).. For deep learning model, GloVe pre-trained language model (Pennington et al., 2014) is utilized as word embedding layer. GloVe is publicly available pre-trained model that consists of 6 billion tokens with 400000 vocabulary size. For vector representation of textual app reviews, this paper make use of deep neural language model i.e. BERT with 768 dimension. In our technical setup, all the experiments with BERT and deep learning were carried out in jupyter notebooks. An Intel Xeon E5-2690 CPU with 16 GB of memory was used in the Microsoft windows Server 2012 R2 communication systems. For our experiments with BERT, we use BERT related inspired libraries that help carry out deep learning using BERT pre-trained models and relies on transformers. On the basis of average length of reviews (shown in table 1) we fixed the size of reviews by selecting padding length 30 for shorter reviews (dataset 1, dataset 2, dataset 3 and dataset 5) and 90 for longer review dataset (dataset 4). The shorter reviews are filled with 0, and longer are truncated to fixed length. The dataset is split into training and testing set with 70:30 ratio, which indicate that model is trained on 70% set and tested on 30%. Then, BiLSTM with 100 and 150 memory units are employed respectively in sequential layer of shorter and longer reviews. In the CNN layer, filter and kernel size are set to 32 and 3 respectively. The global max pooling is used. The hidden layers are associated with RELU activation function set to 16. The dense layer is added with dropout coefficient 0.2. The output layer used softmax activation function for multi-class classification. Adam optimizer was used as optimization algorithm with learning rate of 1e-2 with decay rate of 0.5. The Batch_size for short and long review dataset is 16 and 32 respectively. Extensive experiments are carried on 10-cross validation technique to remove the biasness from the results. The hyperparameters setting for experimental evaluations have been presented in table 7 (section 5.7).

The empirical evaluation of proposed framework is performed on five functional app reviews based datasets. Prior research works evaluated the state-of-the-art deep learning architecture on aforementioned datasets. The main objective of this paper is to investigate the role of BERT as word embedding layer on textual app reviews. The development of hybrid deep neural networks for app review classification is promising research directions. In recent research works, on functional app review classification, hybrid models reported remarkable results. By taking inspiration from existing work, this research work also combine two deep neural networks i.e. CNN and RNN. For comprehensive analysis, eleven baseline models are developed for app review classification.

5.3 Experiment measure

Four evaluation measures such as Precision, Recall, F1-score and Accuracy are employed to measure the performance of the model. Moreover, these evaluation measures are utilized by existing research works. These evaluation measures are formulated as follows:

$$Precision = \frac{T_P}{T_P + F_P} \tag{14}$$

$$Recall = \frac{T_P}{T_P + F_N} \tag{15}$$

$$F1 - measure = \frac{2 \times Pre \times Rec}{Pre + Rec} \tag{16}$$

$$Accuracy = \frac{T_P + T_N}{T_P + F_P + T_N + F_N} \tag{17}$$

Where, T_P (True Positive) indicates that model predict the actual label of the instance; F_P (false positive) and F_N (false negative) indicate that model predict incorrect label other than actual label.

5.4 Baseline Models

In this section, various deep learning methods are used for multi-class classification problems. Deep learning methods such as Artificial Neural Network (ANN), Deep Convolutional Neural Network (DeepCNN), TextCNN, TextRNN, TextRCNN, TextRNN-Att, LSTM, Transformer, BERT, BERT-CNN, and BERT-RNN are considered for comparison. The proposed research work selects only deep learning models; comparative analysis of machine learning is out of scope of this paper. This research work compared eleven baseline deep learning models with BERT-RCNN for multi-class app review classification problem. The details of these methods are as follows:

- **BERT-CNN:** The model comprised five modules, namely, lexicon encoder, multi-layer transformer encoder, local CNN encoder, transformer encoder and output layer. BERT encoder layer retain the contextual relationship between the encoded texts and CNN layer as task-specific layer employed to get the important feature in the text. The output of CNN layer is fed to transformer encoder to obtain final representation of text (Duan et al., 2019)**.**
- **BERT-RNN:** This model starts with Bidirectional transformer encoder, followed by recurrent neural network. In BERT layer, effective contextual word representation has been extracted from the text. Then,

RNN module as belief tracker has been employed. Finally, softmax layer stacked over RNN to predict the categories of text (Lee et al., 2020).

- **BERT:** The model aims to capture contextual features from the textual reviews. For this aforementioned purpose, BERT model is designed that capture the bidirectional representation of unlabeled texts in its encoder, hidden and attention layers. Then, max-pooling layer is employed, followed by fine-tuning layer to obtain the state-of-the-art results for text classification (Devlin et al., 2019).
- **Transformer:** The model embodied transformer module, followed by feed forward network. In transformer module, multi-head attention mechanism has been utilized to unidirectional feature extraction. Then, feed forward network with softmax layer is employed to obtain state-of-the-art results in text classification (Vaswani et al., 2017).
- **Long-short term memory (LSTM):** The model comprised LSTM layer, followed by the softmax layer. The LSTM memory unit consists of input, forget and output gate used to pass the information and store such information in memory units for further purpose Graves and Schmidhuber,(2005).
- **TextRNN:** The model receives input as vector. These input vectors are fed to LSTM module for feature extraction. Then, softmax layer is employed to correctly classify the text into specific classes (Liu et al., 2016).
- **TextRNN-Att:** The model comprised of Bidirectional LSTM and attention mechanism. Initially, embedding layer converts the sentences into vector matrix, and these vectors are fed to bidirectional LSTM layer for feature extraction. Then, attention mechanism has been utilized to capture the important features. Finally, the fully connected layer with softmax activation function has been utilized for classification purpose (Deng et al., 2021).
- **TextRCNN:** The model comprises embedding layer, bidirectional layer, convolutional layer, max-pooling layer, followed by fully connected layer (Lai et al., 2015).
- **TextCNN:** The model utilized word2vec in embedding layer, and input vectors are embed to CNN. Then, max-pooling followed by fully connected layer have been utilized for text classification (Kim, 2011).
- **DeepCNN:** The model comprised series of convolutional blocks, followed by max-pooling layer (Johnson, 2017).
- **ANN:** In this model, series of dense layer are applied to word vector and results are fed to output layer for classification (Ghiassi et al., 2012).

5.5 Experimental results

5.5.1 Comparison with existing studies

In the analysis of state-of-the-art on automated review classification made by Genc-Nayebi and Abran (2017) identification and categorization of reviews is positioned before maintenance and release planning. This means review identification and categorization is very important in research direction to reduce the gap between app reviews and software requirements. Among existing related works, the most similar one to ours are discussed as follows.

5.5.1.1 SUR-Miner vs BERT-RCNN

Gu and Kim, (2016) have introduced Software User Review Miner (SUR-Miner) for automatic summarization of app reviews. Their semi-supervised machine learning approach incorporated with Maximum entropy to classify these reviews into five categories. The author specifically used topic modeling and part-of-speech tags for analysis of textual reviews. However, there approach needs human intervention for identification of topics of reviews. The authors have manually collected the app reviews of 17 different android apps. However, the app reviews tend to be human language which is considered as informal language. Such approach seems infeasible, whenever app reviews evolve, because topic modelling needs domain knowledge. This is because SUR-Miner seems infeasible whenever the syntactic and semantic structure of the app review changes. This approach increases the cost. To overcome this issue, this research paper automatically classifies app reviews in specified classes. BERT model retain the contextual relationship between the words also maintain the syntactic and semantic structure of the textual reviews. Then, deep learning layers are used to extract more meaningful features from the contextual information. This research evaluates the performance of proposed model for comparison purpose. Fig 4 (a) clearly demonstrates that BERT-RCNN performs better than SUR-Miner in app classification on dataset 1.

5.5.1.2 DeepCNN vs BERT-RCNN

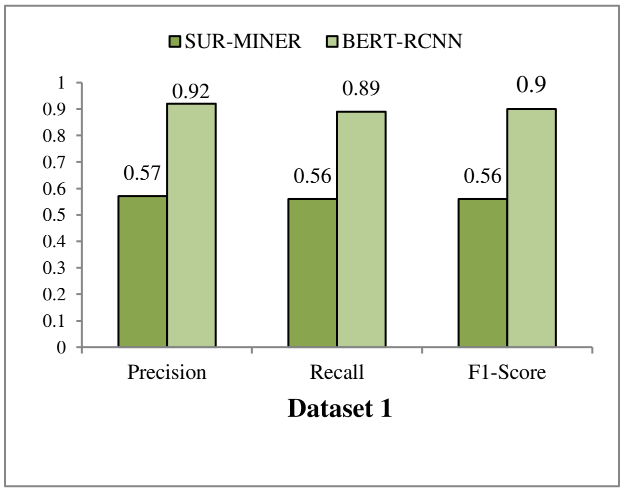
Stanik et al., (2019) have developed deep learning model for classification of user feedback. The authors have sampled 10,000 English reviews from android apps. These crowd-sourced reviews are labeled by two human annotators. In their model, FastText pre-trained language model is used to generate word vector, it also capture the spelling mistake. Then, convolutional layer is used to capture the local features from vectors obtained from previous layer. Their approach yields 0.78, 0.93, and 0.85 of precision, recall and F1-measure respectively. However, the proposed approach applied BERT-RCNN approach and achieves 0.90, 0.87 and 0.88 of precision, recall, and f1-score respectively. In addition, DeepCNN obtained high recall value than precision, which indicate that their approach predict more false positive values. The comparative analysis of both techniques on same dataset is shown in fig 4 (b).

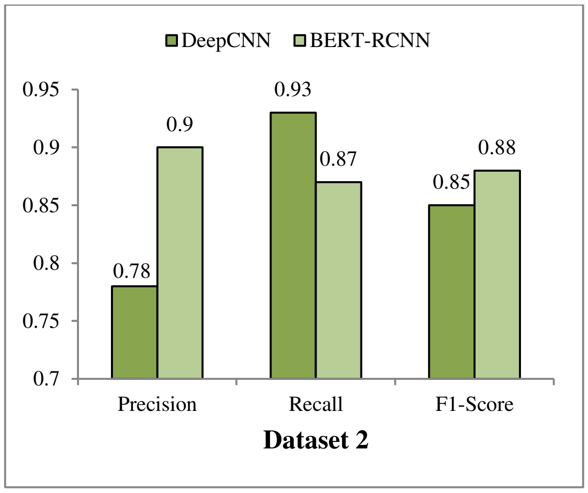
5.5.1.3 AUR-BOW vs BERT-RCNN

Lu and Liang, (2017) have collected app reviews from iBooks and whatsapp. The authors have created ground truth dataset by randomly sampling the reviews and manually classified them into specified classes. They have designed classification technique by combining machine learning algorithms. Word2vec technique is used for training the model, and reported that Naïve Bayes yields satisfactory results with 0.72 of precision, 0.65 of recall and 0.68 of f1-score. However, this paper evaluated the performance of proposed model on the same dataset and obtained 0.96, 0.87, and 0.91 of precision, recall and F1-score respectively. The visualization of performance of both techniques is shown in fig 4(c).

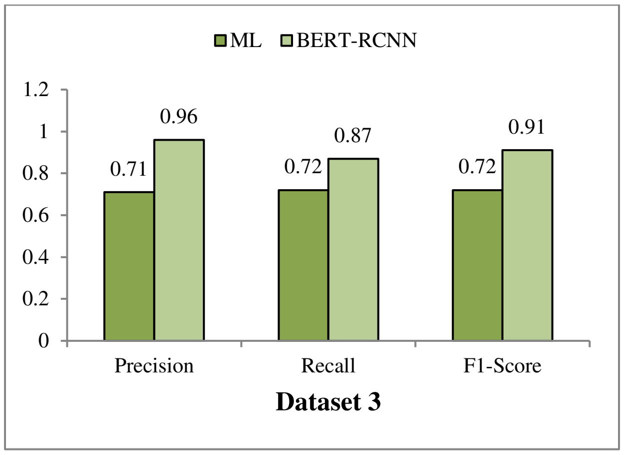
5.5.1.4 Machine learning vs BERT-RCNN

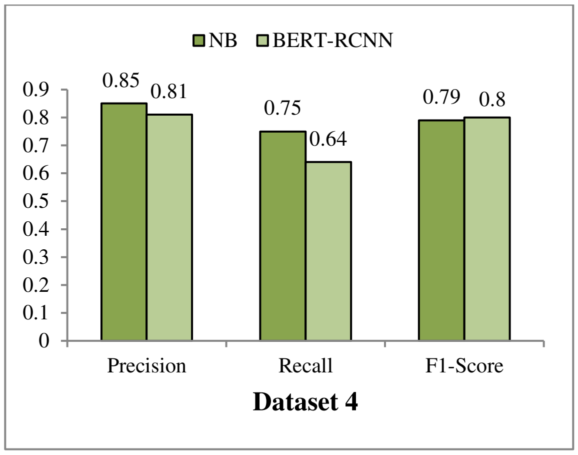
Maalej et al., (2016) have utilized machine learning techniques for categorize app reviews into Bug reports, feature requests, user experiences, and ratings. The authors have collected app reviews from Google play store. Experimental evaluations reported that their approach achieved average precision of 0.85, average recall of 0.75 and F1-score of 0.79. This research work evaluated proposed model on same dataset and obtained competitive performance i.e. 0.81, 0.64, and 0.80 of precision, recall and f1-measure respectively. The graphical representation of comparative analysis of both research works is shown in fig 4(d).



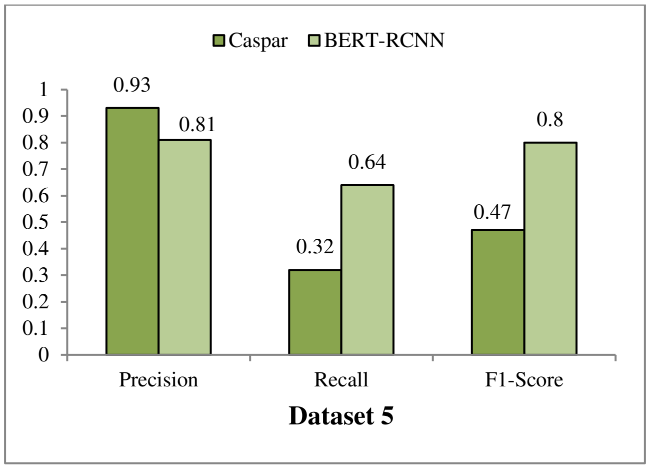


(a) (b)





(c) (d)



(e)

Fig 4: comparative analysis of proposed model with existing approaches

5.5.1.5 Casper vs BERT-RCNN

Guo and Singh, (2020) have presented Casper method for extraction of app problems in user stories. Their model consists of LSTM layer for classification. They evaluated their approach on 1,500 user reviews and yields 92.9 of precision, recall of 32.2 and f1-score of 47.98. Their approach overwhelmed software developers with false positive values. For the comparative analysis, this research paper makes use of same dataset and obtained precision of 0.83, recall of 0.77 and f1-score of 0.80. The visualization of comparative analysis is shown in fig 4(e).

5.5.2 Comparison with baseline methods

In this section, standard deep learning models such as ANN, CNN and RNN and its variants TextRNN, TextCNN, DeepCNN, and TextRCNN are being considered for comparison purpose. Neural language based model are found to be more effective than deep learning models. LSTM and CNN are utilized as the fine-tuning layers to enhance the predictive performance of the model. For the classification of app reviews, researchers have explored the various ensemble architectures and reported results reflect the application of different techniques. In recent years, transfer learning based models carried out satisfactory performances in natural language processing including sentiment analysis and classification etc.

Comparative analysis of proposed model with other eleven state-of-the-art models is presented in table [2-6]. In addition, to investigate the efficiency of BERT-RCNN, these aforementioned state-of-the-art models are compared on the basis of precision, recall, f1-score and accuracy. The dataset (dataset 3) have longer and shorter reviews (dataset 1, dataset 2, dataset 3, and dataset 5). From table [2-6], BERT-RCNN achieves better performance than other benchmark studies. The accuracies of BERT-RCNN are 89%, 89%, 94%, 80%, and 76% for dataset 1, 2, 3, 4, and 5 respectively. It is clearly found that proposed model performs better in all metrics, BERT-RCNN significantly fine-tuned BERT layer by adding layers. This demonstrates the effectiveness of fine-tuning layers on BERT for app review classification and also shows the effect of capturing contextual information better than baseline models. Among, the baseline models except ANN, all models yield satisfactory performance. In addition, conjunction of RNN with CNN significantly improves the performance of BERT model. It shows that proposed model can better extract features with these fine-tuning layers. This is mainly benefitted from (i) BERT model that capture complete semantic information of word and retain the contextual relationship between words. (ii) BERT model is improved by adding RNN layers; because Bidirectional LSTM can hold and extract useful information (iii) local feature in this useful information is extracted by CNN layers. In order to better understand the effect of RCNN layers on BERT models. The confusion matrix shows the misclassified sentences/reviews. To show the performance of proposed BERT-RCNN on five datasets, we have shown the obtained true positive, false positive, true negative and false negative for all datasets. From the figures [5-9], it is found that the highlighted cells after fine-tuning have large number of true positive which indicate that it is better to fine-tuned BERT model for review classification. BERT-CNN and BERT-RNN performs still better than deep learning architectures. It is demonstrated that deep learning models in integration with BERT model extract more important features. The proposed approach achieves the highest accuracy against the other state-of-the-art baseline models. Experimental analysis carried out that:

- Precision of the BERT-RCNN model is significantly better with 92%, 90%, and 96% of precision on dataset 1, dataset 2, and dataset 3 respectively. Precision on dataset 4 is 81% and dataset 5 is 83%.
- The proposed approach yield highest recall values. On dataset 3, BERT model achieves comparable precision to BERT-RCNN, but low recall value. This indicate simple BERT model predict incorrect labels with respect to the training labels. Unlike other algorithms, which shows high recall with low precision, proposed model achieves high precision and high recall on all five datasets.
- F1-score of BERT-RCNN is highest on all the dataset and BERT achieves comparable performance on dataset 3.
- In conclusion, proposed model performs better than the other models on all five datasets.

The visualization of true positive and true negative can be seen on confusion matrix fig [5-9].

For dataset 1, the performance of baseline models have been compared on the basis of precision, recall, f1-measure and accuracy. BERT-RCNN yield better results than other deep learning model. Other neural language model achieves competitive performance in terms of f1-measure and accuracy. From table 2, it can be found that, the utilization of BERT model as pre-trained model achieved higher predictive performance.

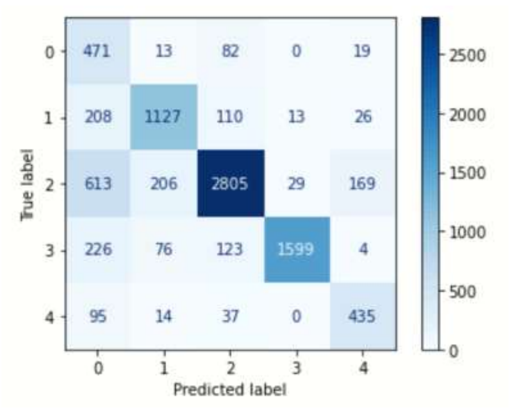
Confusion matrix can also be named as error matrix is used to evaluate the performance of supervised (deep learning and machine learning) techniques. Different instances are represented in confusion matrix with different number of actual and predicted values. Fig 5, the BERT model and proposed model (namely, BERT-RCNN) are compared on the basis of predicted instances. The proposed model is capable to detect the class of the reviews with lower number of false positive. For example, BERT model predict 471 true positive instances with 114 false positive instances, whereas BERT-RCNN predict 489 true positive instances with 98 false positive instances.

Table 2: comparative analysis of proposed model and baseline models on Dataset 1

Architecture	Precision	Recall	F1-measure	Accuracy
ANN	0.44	0.34	0.33	0.58
DeepCNN	0.66	0.60	0.62	0.71
TextCNN	0.70	0.65	0.66	0.73
TextRNN	0.72	0.58	0.61	0.71
TextRCNN	0.73	0.59	0.63	0.72
TextRNN-Att	0.73	0.61	0.65	0.72
LSTM	0.70	0.61	0.63	0.72
Transformer	0.70	0.54	0.58	0.69
BERT	0.83	0.70	0.75	0.73
BERT-CNN	0.84	0.78	0.81	0.80
BERT-RNN	0.86	0.77	0.81	0.81
BERT-RCNN (Proposed approach)	**0.92**	**0.89**	**0.90**	**0.89**

Experimental evaluation on Dataset 2 is presented in table 3, unlike Dataset 1, all the baseline models obtained satisfactory results except ANN. The proposed model yield higher predictive performance with 90% of precision, 87% of recall and 88% of f1-measure. As can be observed in table 3, the accuracy of RNN and CNN seem to have very close performance results with utilization of BERT. Fig 6, proposed model detect instances with lower number of false positive as compared to simple BERT model.

In table 4, precision, recall, f1-measure and accuracy values for Dataset 3 are obtained from baseline models. The empirical findings observed that BERT-RCNN has achieved performance improvement than state-of-the-art models in terms of precision, recall, f1-score and accuracy. Fig 7 represents the true positive instances detected by the proposed model and the simple BERT model.



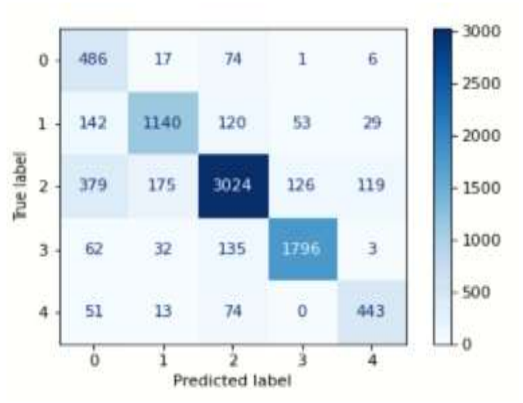


Fig (a): BERT **(b) BERT-RCNN**

Fig 5: confusion matrix for BERT and BERT-RCNN for Dataset 1

Table 3: comparative analysis of proposed model and baseline model on Dataset 2

Architecture	Precision	Recall	F1-measure	Accuracy
ANN	0.48	0.44	0.44	0.62
DeepCNN	0.79	0.81	0.80	0.84
TextCNN	0.80	0.77	0.78	0.83
TextRNN	0.80	0.82	0.80	0.84
TextRCNN	0.81	0.80	0.81	0.84
TextRNN-Att	0.76	0.73	0.74	0.81
LSTM	0.79	0.79	0.79	0.83

Transformer	0.78	0.77	0.77	0.81
BERT	0.86	0.82	0.84	0.85
BERT-CNN	0.84	0.85	0.84	0.87
BERT-RNN	0.84	0.84	0.84	0.85
Proposed approach	**0.90**	**0.87**	**0.88**	**0.89**

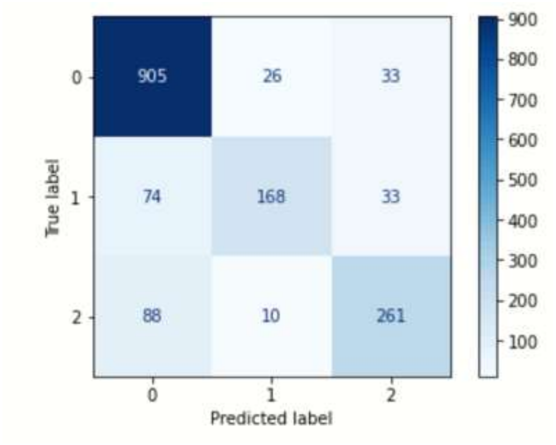
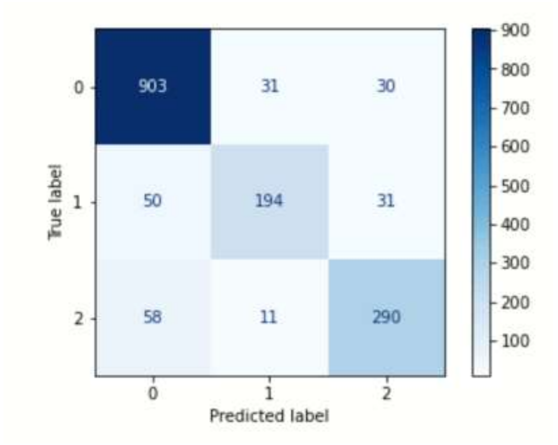


Fig (a): BERT

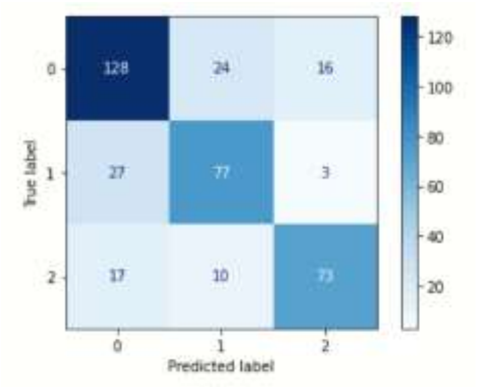


(b) BERT-RCNN

Fig 6: confusion matrix for BERT and BERT-RCNN for Dataset 2

Table 4: comparative analysis of proposed model and baseline model on Dataset 3

Architecture	Precision	Recall	F1-measure	Accuracy
ANN	0.27	0.27	0.25	0.41
DeepCNN	0.67	0.62	0.62	0.72
TextCNN	0.63	0.62	0.61	0.68
TextRNN	0.72	0.58	0.57	0.69
TextRCNN	0.67	0.53	0.55	0.73
TextRNN-Att	0.58	0.59	0.57	0.71
LSTM	0.86	0.47	0.47	0.69
Transformer	0.66	0.50	0.52	0.68
BERT	0.91	0.55	0.60	0.73
BERT-CNN	0.83	0.71	0.76	0.81
BERT-RNN	0.85	0.72	0.77	0.80
BERT-RCNN(Proposed approach)	**0.96**	**0.87**	**0.91**	**0.94**



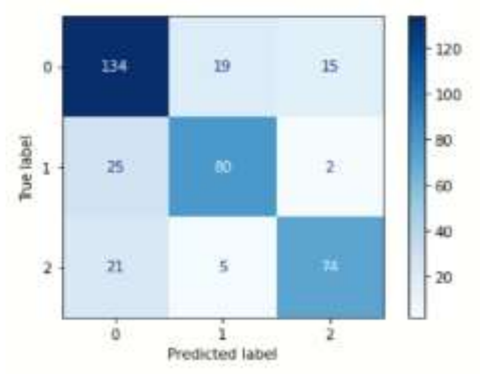


Fig (a): BERT **(b) BERT-RCNN**

Fig 7: confusion matrix for BERT and BERT-RCNN for Dataset 3

The experimental evaluation in Dataset 4 is presented in table 5. The proposed model achieves high predictive performance with 81%, 64%, 69% and 80% of precision, recall, f1-measure and accuracy. Unlike other datasets, other deep learning (ANN, DeepCNN, TextCNN, TextRNN, TextRCNN, TextRNN-Att, and LSTM) and neural language models (BERT, BERT-CNN, and BERT-RNN) obtained satisfactory performance in classification. From fig 8, simple BERT model is incapable of detecting instance of one class, whereas, after fine-tuning the BERT model, fifteen instances are correctly classified. In addition, classification accuracy can also be improved by decreasing the number of false positive instances.

Table 5: comparative analysis of proposed model and baseline model on Dataset 4

Architecture	Precision	Recall	F1-measure	Accuracy
ANN	0.32	0.30	0.30	0.59
DeepCNN	0.45	0.44	0.44	0.66
TextCNN	0.48	0.43	0.43	0.68
TextRNN	0.54	0.44	0.45	0.70
TextRCNN	0.54	0.42	0.45	0.70
TextRNN-Att	0.51	0.43	0.41	0.71
LSTM	0.49	0.42	0.41	0.71
Transformer	0.40	0.40	0.39	0.70
BERT	0.58	0.31	0.34	0.62
BERT-CNN	0.62	0.41	0.45	0.63
BERT-RNN	0.56	0.51	0.53	0.66
BERT-RCNN (Proposed approach)	**0.81**	**0.64**	**0.69**	**0.80**

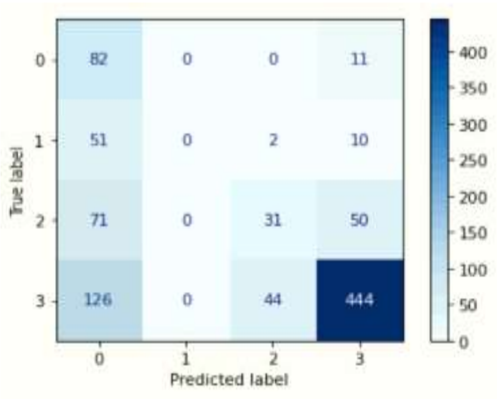


Fig (a): BERT

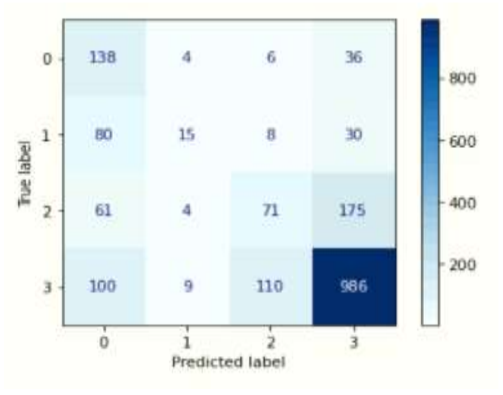

(b) BERT-RCNN

Fig 8: confusion matrix for BERT and BERT-RCNN for Dataset 4

Table 6 shows the performance of baseline models and the proposed model. Empirical evaluation indicates that simple BERT model and BERT-RNN achieves competitive performance with the proposed model. Fig 9 shows that proposed model predict true instance with lower number of false positive for two classes.

Table 6: comparative analysis of proposed model and baseline model on Dataset 5

Architecture	Precision	Recall	F1-measure	Accuracy
ANN	0.43	0.38	0.32	0.47
DeepCNN	0.63	0.58	0.59	0.62
TextCNN	0.61	0.57	0.58	0.59
TextRNN	0.62	0.59	0.60	0.62
TextRCNN	0.65	0.58	0.59	0.63
TextRNN-Att	0.44	0.41	0.34	0.51
LSTM	0.69	0.61	0.63	0.64
Transformer	0.58	0.60	0.58	0.58
BERT	0.81	0.71	0.75	0.73
BERT-CNN	0.80	0.67	0.72	0.69
BERT-RNN	0.78	0.73	0.76	0.74
BERT-RCNN(Proposed approach)	**0.83**	**0.77**	**0.80**	**0.76**

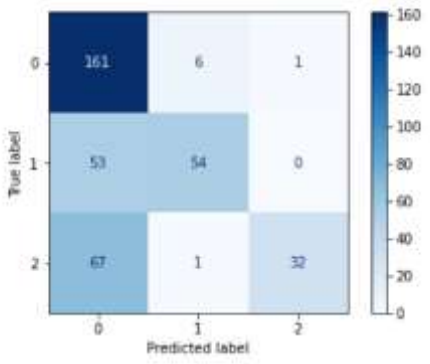


Fig (a): BERT

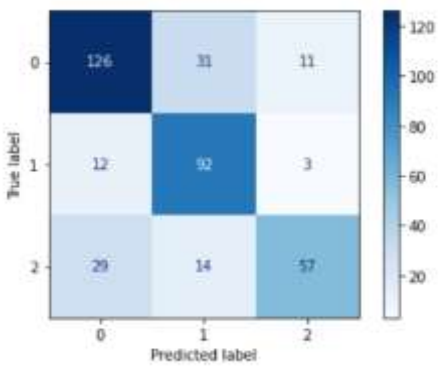

(b) BERT-RCNN

Fig 9: confusion matrix for BERT and BERT-RCNN for Dataset 5

5.6 Hyperparameter tuning

For hyperparameter training, k-cross validation technique is utilized. Some of the parameters remained unchanged. For example, since the addressed problem is multi-class classification problem, categorical cross-entropy is used as a loss function. The Adam (Adaptive moment optimizer) is used for optimization, also utilized to avoid the unstable training or failure in the training model. Table 7 presents the different parameter for five datasets.

Table 7: The best hyperparameter configuration

Parameter	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Embedding dimension	728	728	728	728	728
Maximum sequence length	30	30	30	90	30
BiLSTM unit	100	100	100	150	100
CNN filter	32	32	32	32	32
CNN kernel size	3	3	3	3	3
Padding size	30	30	30	90	30
Optimization	Adam	Adam	Adam	Adam	Adam
Learning rate	0.2	0.2	0.2	0.7	0.2
Hidden layer	32	32	32	64	32
Weight decay	1e-5	1e-5	1e-5	2e-5	1e-5
Number of output layers	5	3	4	4	3
Output layer	Softmax	Softmax	Softmax	Softmax	Softmax
Batch size	16	16	16	32	16

One of the parameter that needs to be tuned is the maximum length of the input text to the models. The average length of dataset 1, dataset 2, dataset 3 and dataset 5 have 8.095, 23,09, 13.46 and 6.015 respectively and dataset 5 have average length of words is 101.75. Selecting the maximum length sentence increases the computational cost and raises several performances issues. Therefore, this proposed research work select 30 as word length for the dataset 1, dataset 2, dataset 3 and dataset 5 and 90 for dataset 5. In most cases, the first few words of a sentence will give enough intuition about the category. For example, the review statement "*Use to love this app but it's not working after new update. Pages won't scroll up or down...none of the different tabs work...it's frozen! Please fix ASAP!!!"* The review sentence was truncated by taking 90 characters as "*Use to love this app but it's not working after new update. Pages won't scroll up or down."* So the modified sentence still classified as the "Bug" category.

Extensive experiments are implemented to obtain the reasonable values of BiLSTM memory units, and to investigate the effect of these memory units. The set of experiments are conducted to investigate the effect of BiLSTM units. During the experiments, the value of memory units varies from 25 to 150. . Except the units, all other parameters are unchanged. From the fig 10, the accuracy of the model is significantly influenced by the BiLSTM memory unit. For the performance of BiLSTM, accuracy continues to improve when the number of neuron increases from 25 to 100 for small length review datasets (such as dataset 1, dataset 2, dataset 3 and dataset 5) and 25 to 150 for dataset 4. They reach the optimal point when the number of neuron is 100 and 150 respectively. Accuracy continues to decrease after reaching at the optimal point for both small length dataset and large text reviews dataset. For BiLSTM, with the increasing number of neurons complexity of model also increases.

The main parameters of convolutional layer are CNN filters and kernel size. Thus, to obtain the kernel size and filters, experiments are conducted on dataset 1 with varying sizes. This paper considers different filter and kernel value. Fig 11 shows the graphical representation of experimental results. From the fig 11, accuracy of BERT-RCNN with filter value 32 is better than with respect to kernel size 3. Moreover, higher accuracy is achieved with kernel size 3 for filter value 32.

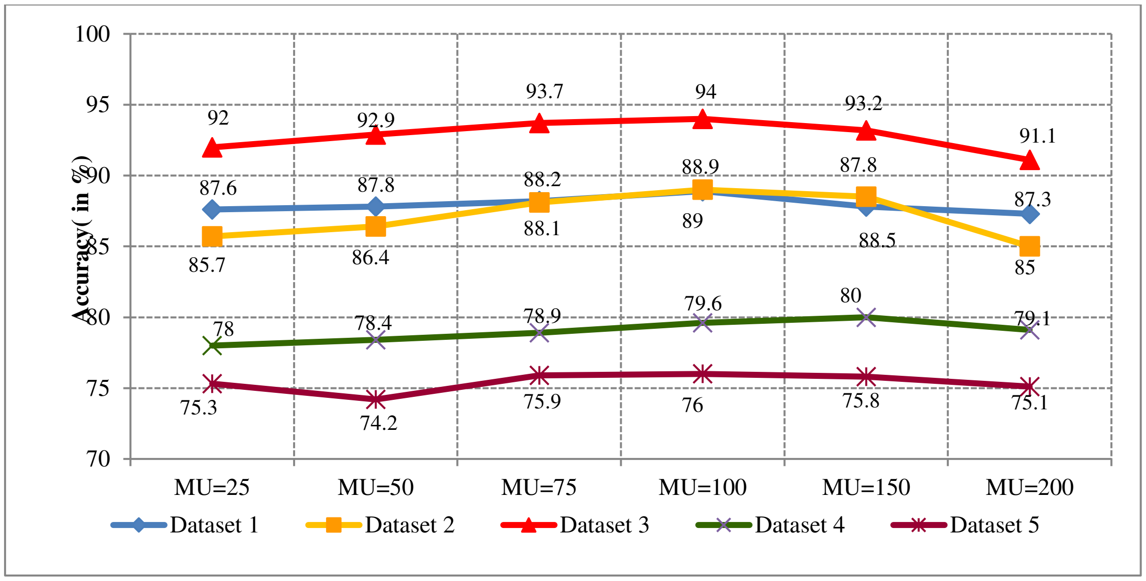


Fig 10: the effect of different memory units

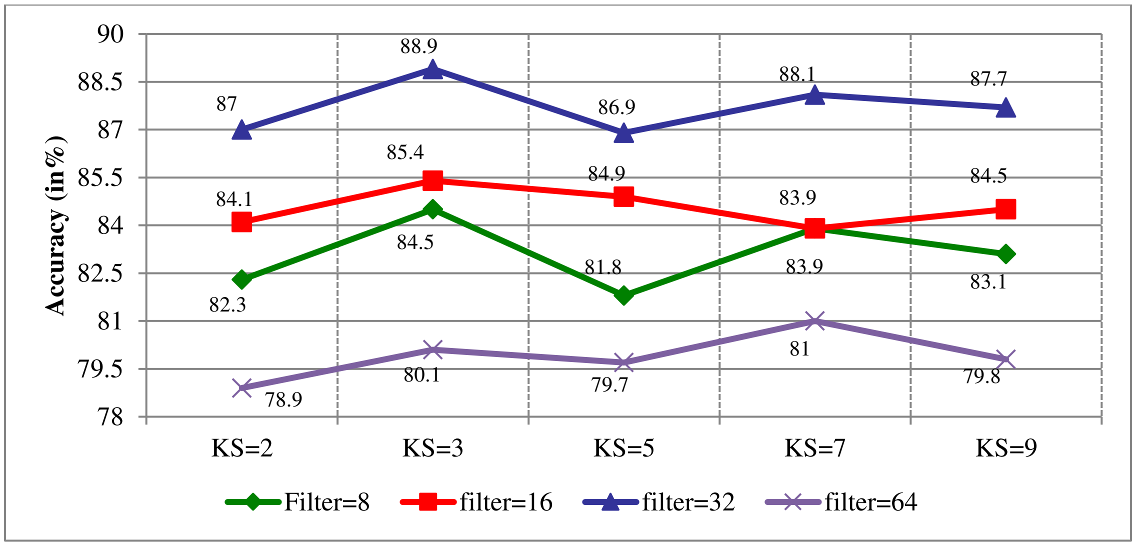


Fig 11: the effect of different kernel size

5.6 Discussion

Extensive experiments on the eleven state-of-the-art deep learning models, several insights are as follows:

- As can be seen from tables [2-6], other deep learning models such as ANN, LSTM, CNN have utilized GloVe word embedding technique. From the predictive performance of neural language models, BERT model performs better than Glove language model.
- The reported results indicate that the fine-tuning language model can obtained promising results on app review classification. The model contains BERT embedding layer to capture the contextual features, Bi-LSTM layer store the contextual information of these features and convolutional layer extract the important features for classification. This paper employs BiLSTM and CNN to extract local and global features. In

this regard, this research work compared the fine-tuned BERT model with the simple BERT model and experiment analysis indicate that BERT-RCNN model can outperform the simple BERT model.

- To investigate the validation, proposed model is compared with eleven recently presented deep neural network models for text classification on five app review datasets. The highest predictive performances have been achieved by BERT-RCNN with classification accuracy of 89% on Dataset 1, a classification accuracy of 89% on Dataset 2, a classification accuracy of 94% on Dataset 3, a classification accuracy of 80% on Dataset 4, and 76% on Dataset 5. The highest predictive performance achieved on Dataset 3 has been obtained by BERT-RCNN with classification accuracy of 94%, a precision value of 0.96, a recall value of 0.87 and F-measure value of 0.91. The second highest predictive performance on Dataset 3 has been obtained by BERT-CNN model with classification accuracy of 81%, a precision value of 0.83, recall value 0.71 and F-value of 0.76. Hence, the percentage increase the rates obtained by the proposed scheme for the Dataset 3 is 14%.
- From the predictive performance of different DNN-based architectures, the experimental evaluations indicate that integration of BiLSTM with CNN layer can yield promising results to capture the long-term dependencies and global semantic features.
- BERT-RCNN can yields high predictive performance results in both short and long review datasets. The predictive performances of deep neural networks are generally higher when the numbers of instances in training sets are more. The deep learning model requires a huge amount of data for proper training and is computationally intensive too.
- The utilization of CNN layer with different filter sizes can learn local features effectively.

5.7 Practical Implication

The research paper proposed a novel framework to classify app reviews using transfer learning. Various machine learning and deep learning methods have been applied for classification; little comparative research has been done on the app review analysis from natural language perspective. Prior studies proposed transfer learning and ensemble models for app review classification with English text, but fine-tuning layers are not added to their model. Combining BERT layer, BiLSTM layer, and CNN layer, our proposed app review classification model not only improves the accuracy of classification as compared to baseline models, but also investigates the important hyperparameters based on the experiments. Therefore, the proposed model can considered as more powerful app review classification models for natural language app reviews.

Online user reviews are considered as rich source of information for both users and developers. This research work indicates that transfer learning and deep learning based multi-class classification models can be utilize in extracting software requirements also Due to the rapid growth in feedback provided by online platforms, and it is dynamic nature, relaying on transfer learning model with additional deep learning classification models can be considered as good option. Moreover, the topic modeling approaches may not always work; because of the informal structure and grammatical errors present in user reviews. The proposed BERT-RCNN framework automatically classifies multi-class software requirements without using preprocessing techniques make it more capable for RE practices. Automatic classification of quality attribute into respective classes can improve the software quality. Therefore, extracted relevant information can be used for next release planning. To standout in market competition, mobile app developers need to consider the important features for customers identified as needs, expectations, interests and requirements. Recent studies reported that these reviews can be seen from two perspectives. First is from users' perspective, such feedback influences their initial thoughts on whether the app is worth purchasing. From app developers' perspective, user feedback contains technical information that can be useful for planning new feature for next release. Positive user reviews attract more customers and bring financial gains, also in turns analysed to develop better application updates. Despite from this, negative reviews can be potentially misleading and often causes sales loss. Such information can be easily utilized by the developers who are just initiating their business.

6 Threats to validity

6.1 Internal validity

A potential threat to internal validity of proposed approach is in experimental datasets used in this paper. The human annotator was created this dataset with the approach in mind. Therefore, there might be the risk of bias. We mitigated this threat by applying stratified kfold cross validation to avoid the bias, and tested the model on the data samples that had not been revealed during training. However, the instances of the reviews might not be general representatives of all kind of reviews. Moreover, the datasets that we used have some further issues regard to internal validity. Some reviews may be incorrectly labeled and selection of review for the dataset is bias. This results in imbalanced datasets that miss out some kinds of reviews. In addition, other major external threat to the proposed model is the dataset. The reviews were and encoded by human annotators and thus might not illustrate industry standards. Therefore, conclusions on the generalizability based on the dataset are not warranted. We used these datasets as they are used in previous published research work and has its validity and also let us to directly compare our results to existing research works.

Another threat to internal validity is tuning of hyperparameter. The BERT model has large number of hyperparameters that can be tuned to potentially improve the performances. These sets of hyperparameters are attention mechanism, number of hidden layer, batch size, and dropout size etc. Tuning all hyperparameters infeasible in practice due to computational cost and GPUs required by transformer based model. To mitigate this risk, only memory unit, kernel size and filters are tuned in this research work.

6.2 External validity

One potential threat to our external validity is the dataset used in our research work. The experimental datasets are created by considering approach in mind. 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 study the ability of pre-trained language model in requirements classification of app reviews. The app reviews classification mainly focuses on extracting useful information in 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 do not study pre-trained language model for other purposes such as finding issues in business perspective, product reviews, and application such as attention mining.

6.3 Conclusion validity

A threat to the statistical conclusion validity might occur due to the usage of imbalanced datasets. Here, multi-class datasets have different classes. Unfortunately, undersampling and oversampling techniques still needs more exploration.

6.4 Construct validity

A threat to construct validity is that the labels in exploited dataset could be incorrect. The datasets adopted in this research work manually labeled with the help of human annotators; the labeled dataset could be incorrect for different reasons. As a result, such incorrect labeling could produce inaccurate results.

7. 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 perspective. With development of machine and deep learning models widely exploit in automatic classification of multi-class user reviews. These existing models have some limitations; also the classification accuracy of these models can be enhanced. This paper presents a hybrid BERT-RCNN model for app review classification. Firstly, this model utilizes BERT as word embedding model for textual app reviews. This word representation model extracts the contextual relationship between the words. Next, this contextual information is fed to the BiLSTM layer to capture the sequential features, and then CNN utilized to reduce the dimensionality of data. It is worth mentioning that BERT model can be effectively fine-tune with RNN and CNN layers. To investigate the effectiveness of proposed model, extensive experiments are performed on the five benchmark datasets. The experimental results evidently prove the effectiveness of BERT-RCNN in identifying the important issues in app reviews. In addition, hyperparameter such as kernel size, filter value, and BiLSTM unit is tuned to improve the overall classification of proposed model.

Comparing our approach to some existing related work (in section 5.5.1), we can find that our BERT-RCNN methods is more effective in classification quality and recognize more semantic information of words than other approaches discussed in literature. It is worth noting that applying techniques for classification are the promising direction for mining software requirements from app reviews. Thus, we are confident that our proposed model will eventually leads to practice-oriented tools for identification and classification of quality and relevant requirements from large amount of user feedback. From this research paper, it is observed that in RE, it is worthwhile to explore app stores as an information channels, specifically when analyzing feedback to support in accounts of software companies.

8. Limitations and Future directions

Due to rapid growth of social media, millions of data is produced due to its gained popularity. Thus, manual analysis of such data seems infeasible for the researchers. Therefore, there is urgent need for more efficient methods for classification of user's opinion. Existing classification models are often unable to meet the demand of real-world big data processing especially for large-scale datasets. Therefore, developers still need model for automatic identification and classification of app reviews.

Although, our proposed model can improve the accuracy of classification model, it has limited capability to predict the class of some specific context, such as multi-label app reviews. For future work, it is vital to develop more efficient neural network model to solve these problems. From this research work, following research directions can be carried out; (i) using other neural language model for app review classification. (ii) Using hybrid model with attention mechanism in multi-label reviews. (iii) Develop multi-lingual based model for app review classification such as Punjabi and Hindi etc. We also envisage future work that combines users' feedback from multiple crowdsourced platforms such as Twitter, Amazon and User forums etc for obtaining important information, ideas and inspiration about how to improve a software product.

Contribution Statement

Kamaljit Kaur: Methodology, Software, Formal Analysis, Investigation, Writing-original draft, visualization. **Parminder Kaur:** Conceptualization, Methodology, Validation, Formal Analysis, Investigation, Writing- Review and editing, Supervision.

Data Availability

The data of the study is private. All the datasets are collected from respected creators.

Conflict of Interest

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

Funding Declaration

No funding was received for conducting this study.

References

Aslam, N., Ramay, W. Y., 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

Basiri, M. E., Nemati, S., Abdar, M., Cambria, E., & Acharya, U. R. (2021). ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis. Future Generation Computer Systems, 115, 279-294.

Campbell, J. C., Hindle, A., & Stroulia, E. (2015). Latent Dirichlet Allocation: Extracting Topics from Software Engineering Data. *The Art and Science of Analyzing Software Data*, *3*, 139–159. https://doi.org/10.1016/B978-0-12-411519-4.00006-9

Chen, N., Lin, J., Hoi, S. C. H., Xiao, X., & Zhang, B. (2014). AR-miner: Mining informative reviews for developers from mobile app marketplace. *Proceedings - International Conference on Software Engineering*, *1*, 767–778. https://doi.org/10.1145/2568225.2568263

Chollet F (2021) Deep learning with Python. Simon and Schuster;

Deng, J., Cheng, L., & Wang, Z. (2021). Attention-based BiLSTM fused CNN with gating mechanism model for Chinese long text classi fi cation. *Computer Speech & Language*, *68*, 101182. https://doi.org/10.1016/j.csl.2020.101182

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*(Mlm), 4171–4186.

Duan, L., Hou, J., Qiao, Y., & Miao, J. (2019). Epileptic Seizure Prediction Based on Convolutional Recurrent Neural Network with Multi-Timescale. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 11936 LNCS*. https://doi.org/10.1007/978-3-030-36204-1_11

Fu, B., Lin, J., Liy, L., Faloutsos, C., Hong, J., & Sadeh, N. (2013). Why people hate your App - Making sense of user feedback in a mobile app store. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Part F128815*, 1276–1284. https://doi.org/10.1145/2487575.2488202

Gao, C., Zeng, J., Lyu, M. R., & King, I. (2018). Online App Review Analysis for Identifying Emerging Issues. *Proceedings - International Conference on Software Engineering*, *2018-January*, 48–58. https://doi.org/10.1145/3180155.3180218

Ghiassi, M., Olschimke, M., Moon, B., & Arnaudo, P. (2012). Expert Systems with Applications Automated text classification using a dynamic artificial neural network model. *Expert Systems With Applications*, *39*(12), 10967–10976. https://doi.org/10.1016/j.eswa.2012.03.027

Gu, X., & Kim, S. (2016). What parts of your apps are loved by users? *Proceedings - 2015 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015*, 760–770. https://doi.org/10.1109/ASE.2015.57

Guo, H., & Singh, M. P. (2020). Caspar: Extracting and synthesizing user stories of problems from app reviews. *Proceedings - International Conference on Software Engineering*, 628–640. https://doi.org/10.1145/3377811.3380924

Guzman, E., & Maalej, W. (2014). How do users like this feature? A fine grained sentiment analysis of App reviews. *2014 IEEE 22nd International Requirements Engineering Conference, RE 2014 - Proceedings*, 153–162. https://doi.org/10.1109/RE.2014.6912257

Hadi, M. A., & Fard, F. H. (2020a). AOBTM: Adaptive Online Biterm Topic Modeling for Version Sensitive Short-texts Analysis. *Proceedings - 2020 IEEE International Conference on Software Maintenance and Evolution, ICSME 2020*, 593–604. https://doi.org/10.1109/ICSME46990.2020.00062

Hadi, M. A., & Fard, F. H. (2020b). *ReviewViz: Assisting Developers Perform Empirical Study on Energy Consumption Related Reviews for Mobile Applications*. 1–4. https://doi.org/10.1145/3387905.3388605

Haering, M., Stanik, C., & Maalej, W. (2021). Automatically matching bug reports with related app reviews. *Proceedings - International Conference on Software Engineering*, 970–981. https://doi.org/10.1109/ICSE43902.2021.00092

He, D., Hong, K., Cheng, Y., Tang, Z., & Guizani, M. (2019). Detecting Promotion Attacks in the App Market Using Neural Networks. *IEEE Wireless Communications*, *26*(4), 110–116. https://doi.org/10.1109/MWC.2019.1800322

Henao, P. R., Fischbach, J., Spies, D., Frattini, J., & Vogelsang, A. (2021). Transfer Learning for Mining Feature Requests and Bug Reports from Tweets and App Store Reviews. *Proceedings of the IEEE International Conference on Requirements Engineering*, *2021-September*, 80–86. https://doi.org/10.1109/REW53955.2021.00019

Hey, T., Keim, J., Koziolek, A., & Tichy, W. F. (2020). NoRBERT: Transfer Learning for Requirements Classification. *Proceedings of the IEEE International Conference on Requirements Engineering*, *2020-Augus*, 169–179. https://doi.org/10.1109/RE48521.2020.00028

Johnson, R., & Zhang, T. (2017). Deep pyramid convolutional neural networks for text categorization. *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers) (pp. 562-570).

Chen, Y. (2015). Convolutional neural network for sentence classification (Master's thesis, University of Waterloo).

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. *Proceedings of the National Conference on Artificial Intelligence*, *3*, 2267–2273.

Lee, H., Lee, J., & Kim, T. Y. (2020). SUMBT: Slot-utterance matching for universal and scalable belief tracking. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, 5478–5483. https://doi.org/10.18653/v1/p19-1546

Liu, P., Qiu, X., & Xuanjing, H. (2016). Recurrent neural network for text classification with multi-task learning. *IJCAI International Joint Conference on Artificial Intelligence*, *2016-January*, 2873–2879.

Lu, M., & Liang, P. (2017). Automatic classification of non-functional requirements from augmented app user reviews. *ACM International Conference Proceeding Series*, *Part F1286*, 344–353. https://doi.org/10.1145/3084226.3084241

Maalej, W., Kurtanović, Z., Nabil, H., & Stanik, C. (2016). On the automatic classification of app reviews. *Requirements Engineering*, *21*(3), 311–331. https://doi.org/10.1007/s00766-016-0251-9

Maalej, W., & Nabil, H. (2015). Bug report, feature request, or simply praise? On automatically classifying app reviews. *2015 IEEE 23rd International Requirements Engineering Conference, RE 2015 - Proceedings*, 116–125. https://doi.org/10.1109/RE.2015.7320414

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. *Empirical Software Engineering*, *21*(3), 1067–1106. https://doi.org/10.1007/s10664-015-9375-7

Nayebi, M., Cho, H., & Ruhe, G. (2018). App store mining is not enough for app improvement. In *Empirical Software Engineering* (Vol. 23, Issue 5). Empirical Software Engineering. https://doi.org/10.1007/s10664-018-9601-1

Oh, J., Kim, D., Lee, U., Lee, J. G., & Song, J. (2013). Facilitating Developer-User Interactions with Mobile App Review Digests. *Conference on Human Factors in Computing Systems - Proceedings*, *2013-April*, 1809–1814. https://doi.org/10.1145/2468356.2468681

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* (Vol. 4, pp. 2047-2052). IEEE.

Qiao, Z., Wang, A., Abrahams, A., & Fan, W. (2020). *Association for Information Systems Association for Information Systems Deep Learning-Based User Feedback Classification in Mobile App Deep Learning-Based User Feedback Classification in Mobile App Reviews Reviews*. 1–13. https://aisel.aisnet.org/sigdsa2020

Rustam, F., Mehmood, A., Ahmad, M., Ullah, S., Khan, D. M., & Choi, G. S. (2020). Classification of Shopify App User Reviews Using Novel Multi Text Features. *IEEE Access*, *8*, 30234–30244. https://doi.org/10.1109/ACCESS.2020.2972632

Scalabrino, S., Bavota, G., Russo, B., Penta, M. Di, & Oliveto, R. (2019). Listening to the Crowd for the Release Planning of Mobile Apps. *IEEE Transactions on Software Engineering*, *45*(1), 68–86. https://doi.org/10.1109/TSE.2017.2759112

Shah, F. A., Sirts, K., & Pfahl, D. (2019). Simple app review classification with only lexical features. *ICSOFT 2018 - Proceedings of the 13th International Conference on Software Technologies*, *Icsoft*, 112–119. https://doi.org/10.5220/0006855901460153

Stanik, C., Haering, M., & Maalej, W. (2019). Classifying multilingual user feedback using traditional machine learning and deep learning. *Proceedings - 2019 IEEE 27th International Requirements Engineering Conference Workshops, REW 2019*, 220–226. https://doi.org/10.1109/REW.2019.00046

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999–6009.

Villarroel, L., Bavota, G., Russo, B., Oliveto, R., & Di Penta, M. (2016). Release planning of mobile apps based on user reviews. *Proceedings - International Conference on Software Engineering*, *14-22-May-*, 14–24. https://doi.org/10.1145/2884781.2884818

Wardhana, J. A., & Sibaroni, Y. (2021). Aspect Level Sentiment Analysis on Zoom Cloud Meetings App Review Using LDA. Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), 5(4), 631-638.

Yang, T., Gao, C., Zang, J., Lo, D., & Lyu, M. (2021). TOUR: Dynamic Topic and Sentiment Analysis of User Reviews for Assisting App Release. *The Web Conference 2021 - Companion of the World Wide Web Conference, WWW 2021*, 708–712. https://doi.org/10.1145/3442442.3458612