Empirical Software Engineering (2023) 28:88
https://doi.org/10.1007/s10664-023-10314-x

# Evaluating pre-trained models for user feedback analysis in software engineering: a study on classification of app-reviews

**Mohammad A Hadi[1] · Fatemeh H. Fard[1]**

Accepted: 3 March 2023 / Published online: 23 May 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

**Abstract**

**Context** Automatic classification of mobile applications users' feedback is studied for different areas of software engineering. However, supervised classification requires a lot of manually labeled data, and with introducing new classes or new platforms, new labeled data and models are required. Employing Pre-trained neural Language Models (PLMs) have found success in the Natural Language Processing field. However, their applicability has not been explored for app review classification.

**Objective** We evaluate using PLMs for issue classification from app reviews in multiple settings and compare them with the existing models.

**Method** We set up different studies to evaluate the performance and time efficiency of PLMs compared to Prior approaches on six datasets: binary vs. multi-class, zero-shot, multi-task, and multi-resource settings. In addition, we train and study domain-specific (Custom) PLMs by incorporating app reviews in the pre-training. We report Micro and Macro Precision, Recall, and F1 scores and the time required for training and predicting with the models.

**Results** Our results show that PLMs can classify the app issues with higher scores, except in multi-resource setting. On the largest dataset, results are improved by 13 and 8 micro- and macro-average F1-scores, respectively, compared to the Prior approaches. Domain-specific PLMs achieve the highest scores in all settings with less prediction time, and they benefit from pre-training with a larger number of app reviews. On the largest dataset, we obtain 98 and 92 micro- and macro-average F1-score (from 4.5 to 8.3 more F1-score compared to general pre-trained models), 71 F1-score in zero-shot setting, and 93 and 92 F1-score in multi-task and multi-resource settings, respectively, using the large domain-specific PLMs.

Communicated by: Raula Gaikovina Kula, Jin Guo, Neil Ernst.

This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada, RGPIN-2019-05175.

This article belongs to the Topical Collection: *Special Issue on Registered Reports*.

✉ Fatemeh H. Fard
fatemeh.fard@ubc.ca

Extended author information available on the last page of the article

**Conclusion** Although prior approaches achieve high scores in some settings, PLMs are the only models that can work well in the zero-shot setting. When trained on the app review dataset, the Custom PLMs have higher performance and lower prediction times.

**Keywords** Pre-trained neural language models · App review classification

## 1 Introduction

Mobile application (app) users can write feedback about their usage experience on social media (Stanik et al. 2019) or app marketplaces (Hadi and Fard 2020). Previous studies used these reviews to identify the app issues and are essential from software engineering perspectives such as requirement engineering, software maintenance, and testing (Kaur and Kaur 2022; Lu and Liang 2017; Ciurumelea et al. 2017; Al-Hawari et al. 2021; Grano et al. 2018). As the manual investigation of the reviews is time consuming, due to a lot of unrelated review texts, multiple researchers developed automatic classification techniques to extract useful information from the users' feedback using rule-based, machine learning, or deep learning approaches (Chen et al. 2014; Gu and Kim 2015; Guzman et al. 2015; Stanik et al. 2019). However, a primary requirement of supervised techniques is the availability of labeled datasets, which needs to be done manually by field-experts (Stanik et al. 2019). Although app review analysis is helpful for software developers in practice, this manual effort is expensive and seems to be a barrier in many studies (Kaur and Kaur 2022).

Whenever the classification purpose (i.e., labels of interest) or platform changes, the models require retraining on newly labeled datasets since they do not retain the natural language properties learned from previous training, including the linguistic features and the context of the trained labels (Maalej and Nabil 2015). Training from scratch prevents the extensibility of the developed models for classifying new issues in practice. A recent established and widely accepted practice in Natural Language Processing (NLP) is using Pre-Trained neural Language Models (PLM) and then transferring its learned knowledge to various downstream NLP tasks, such as sentiment analysis, question answering, or classification (Qiu et al. 2020a). PLMs reduce the amount of effort (i.e., new model development time per task) to build models for each task separately and reduce the amount of required labeled dataset (Liu et al. 2019).

PLMs are used extensively and led to many advances in NLP. Recently, they have also been studied in the software engineering field, such as exploring the use of PLMs for sentiment analysis in software engineering (Zhang et al. 2020; Biswas et al. 2020), API learning (Hadi et al. 2022), mobile app user feedback answer generation (Cao and Fard 2021), and evaluating them in software engineering context (Luong et al. 2022; Von der Mosel et al. 2022) or for tasks related to programming languages such as comment generation (Svyatkovskiy et al. 2020), transfer learning to low resource languages (Chen et al. 2022), assessing comprehensive encoding characteristics of source code (Karmakar and Robbes 2021), and structural analysis of source code (Wan et al. 2022). Nevertheless, the extent to which PLMs can be applied for app review classification still needs to be determined.

Therefore, in this study, we aim to explore the benefits of PLMs compared to the existing approaches for app review analysis, specifically the app review classification tasks. We define the app review classification as the task of extracting useful (i.e., software-engineering-related) information from users' feedback. Our goal is to investigate the accuracy and time efficiency of PLMs for the app issue classification task over different selected datasets from

literature with various labels and multiple tasks (i.e., issue classification and sentiment analysis of app reviews). Therefore, experiments are conducted in different settings: by fine-tuning the PLMs on different sizes of the labeled dataset for downstream tasks, exploring the same PLMs for multiple tasks and resources in app review analysis, and finally, comparing the performance of PLMs when they are trained on non-domain-specific dataset versus PLMs trained on the domain-specific dataset (i.e., app reviews). In all settings, the micro- and macro-average of the precision, recall, and F1-scores (metrics used for classification tasks), as well as the time required to predict the labels, are computed. The definitions of these metrics are detailed in Section 5.4. Our experiments will provide baselines on the applicability of PLMs for app review analysis, including the cost of using them (in terms of required time for predictions) and their capability to reduce the manual effort required to label large datasets.

The contributions of this study are as follows:

- This is the first study that explores the applicability of PLMs for automatic app issue classification tasks compared to the existing tools.
- We extensively compare four PLMs and four existing tools/approaches on seven app review datasets with different sizes and labels.
- We are the first to explore the performance of general versus domain-specific pre-trained models for app review classification.
- This is the first empirical study to examine the accuracy and efficiency of PLMs in four different settings: binary vs. multi-class classification, zero-shot setting, multi-task setting, and setting in which training data is from one resource (e.g., App Store), and we test the models on data from another platform (e.g., Twitter).

We explore the following research questions for our study:

**RQ1: How accurate and efficient are the PLMs in classifying app reviews compared to the existing tools?** In this research question, we explore how the current PLMs perform compared to the existing tools, including their required time for training and prediction. The existing tools are based on curated rules, feature-engineered machine learning algorithms, and deep learning models.

*Findings:* We find that different PLMs outperform Prior approaches by ∼3% to ∼15% on all datasets. Additionally, the best-performing PLMs require slightly more time for prediction, although the difference in prediction time is negligible (+0.04s to +2.87s).

**RQ2: How does the performance of the PLMs change when they are pre-trained on an app-review dataset instead of a generic dataset (e.g., Wiki documents, book corpus)?** The current PLMs are trained using available text scraped from the web. In some studies, domain-specific models are trained, e.g., LEGAL-BERT (Chalkidis et al. 2020), and medical BERT (Wada et al. 2020), which can improve the performance compared to the non-domain-specific models. In RQ2, we explore the performance of a domain-specific pre-trained model when trained on app reviews. We refer to these models as Custom PLMs.

*Findings:* The PLMs trained from scratch on domain-specific data perform better than out-of-the-box models (+0.8% to +2.2% micro-F1 score). The Custom PLMs do not fluctuate much in prediction times with respect to the readily available PLMs. In addition, incorporating more app reviews in the pre-training can help PLMs produce up to 15.2% better micro-F1 score.

**RQ3: How do the PLMs perform in the following settings?**

**(RQ3-1) Binary vs. multi-class setting,**
**(RQ3-2) Zero-shot classification,**
**(RQ3-3) Multi-task setting (i.e., different app-review analysis tasks),**

**(RQ3-4) Classification of user reviews collected from different resources (i.e., Twitter, App Store).**

The answers to this research question will help us understand the applicability of PLMs and their performance in different settings (i) when the classification is on two or more classes; (ii) when a labeled dataset is not available; (iii) the task changes; and (iv) different resources are used. In part (iv), we explore the transferability of the models as the distributions of data on various platforms is different (Ruder and Plank 2017).

*Findings:* Both Prior and PLM models yield better performance for binary classification tasks compared to multi-class settings. PLMs are the best choice for zero-shot classification settings. Custom PLMs improve the results of their non-domain-specific models. For multi-task and multi-resource settings, the Custom LARGE PLMs (i.e., PMTs pre-trained with 10 million app reviews) perform better than readily available PLMs. RoBERTa-based Custom PLMs have the best scores among all models and have lower prediction times in the multi-resource setting than Prior approaches.

**Paper Organization** The rest of the paper is organized as follows. We discuss the background of app review studies and the motivation of our work in Section 2. We review the related works in Section 3. Section 4 discusses the Study Overview. In Section 5 we detail our approach and the experimental setup to answer each of the research questions and discuss the results in Section 6. Section 7 is dedicated to the discussions. Threats to validity are in Section 8, and we conclude the paper in Section 9.

**Data Availability** All the datasets we used for our study are from the literature. Some of them required permission from their authors. Therefore, we cannot release any data with this manuscript. However, the datasets used here are cited, and researchers can contact the publishers of the datasets if needed.

## 2 Background and Motivation

Mobile application (app) marketplaces, such as Apple App Store and Google Play, enable users to rate and review apps (Hadi and Fard 2020). Users express their usage experience by writing reviews from different perspectives, such as the app's quality, performance, and functionality. The reviews provide a distinct way for the application developers to acquire customer feedback (Lu and Liang 2017). The developers monitor the reviews regularly to address users' significant concerns and resolve the reported issues in the forthcoming app update. If the app is not regularly updated addressing the issues, it will gradually lose its popularity (Liu et al. 2015; Islam et al. 2010). Therefore, app reviews have been studied extensively (Kaur and Kaur 2022) and are shown to be resourceful for different intents including requirement engineering (Lu and Liang 2017), release planning (Ciurumelea et al. 2017), software maintenance (Al-Hawari et al. 2021), change-file localization (Zhou et al. 2021), and testing (Grano et al. 2018). Different studies also focused on analyzing the reviews to filter non-informative reviews (Chen et al. 2014), identify bugs (Maalej and Nabil 2015), classify usability and quality concerns (Lu and Liang 2017), and find security and privacy issues (Besmer et al. 2020).

Other than App Stores, mobile app users also express their opinions on social media such as Twitter, which regularly reveals new information for the developers (Stanik et al. 2019). Nonetheless, the manual extraction of informative feedback from App Stores or social media is difficult, as many unrelated and noisy user comments exist. Therefore, researchers have developed techniques, including rule-based, machine learning, or deep learning approaches,

to automatically extract useful information from user reviews to help app developers (Chen et al. 2014; Gu and Kim 2015; Guzman et al. 2015; Stanik et al. 2019). Many studies classify the reviews for various purposes, which range from identifying problems, user inquiries, feature requests, and aspect evaluations (e.g., feature strengths, weaknesses, and performance) to sieving usability, portability, and reliability (Stanik et al. 2019; Kaur and Kaur 2022; Lu and Liang 2017; Ciurumelea et al. 2017; Maalej and Nabil 2015; Besmer et al. 2020). These studies provide app developers with various serviceable information that facilitates making informed decisions for planning app updates. This extra information is crucial for app developers as mobile applications and their updates get released quite frequently over a short period to meet the market requirements; so, it often proves challenging to identify and prune all the software defects and bugs during the testing phase (Joorabchi et al. 2013).

The topics of interest in app review classification are numerous and different studies leverage various supervised machine learning techniques to extract helpful software engineering information from app reviews (Kaur and Kaur 2022). However, the main requirement for training supervised models is access to labeled datasets. Although multiple datasets are open-sourced by the research community (Gu and Kim 2015; Stanik et al. 2019; Lu and Liang 2017; Maalej and Nabil 2015; Guo and Singh 2020), each is restricted to a specific set of classes, such as 'bugs' or 'feature requests.' Therefore, new sets of labeled datasets are required when new classes are the target categories to be classified by the models. For example, to classify the 'security' or 'privacy'-related feedback (Besmer et al. 2020), a dataset with 'feature request' labels might not be appropriate. Recent works try to alleviate the problem by incorporating semi-supervised learning (Deocadez et al. 2017) or active learning (Dhinakaran et al. 2018). However, with the emergence of new labels of interest, developers must label new datasets, which renders the previously curated datasets obsolete. In addition, the distribution of the data collected from different platforms (i.e., App Stores or social media) varies. Therefore, even though the same labels are to be classified from two platforms, a model that has been trained on one (e.g., Google Play) needs to be retrained to extract the same labels from another (e.g., Twitter) to yield similar performance (Stanik et al. 2019).

To address similar issues, a widely studied approach in NLP and recently in Software Engineering is applying PLMs (such as BERT Devlin et al. 2019) to various problems, including classification tasks. PLMs are language models trained on large natural language corpora using a deep neural network in an unsupervised manner (Zhang et al. 2020). These models are then fine-tuned for various downstream tasks using limited labeled datasets. As PLMs are trained on large general domain corpora, they learn contextual linguistic information and eliminate the need to train downstream task models from scratch (Wada et al. 2020), reducing the need for the amount of labeled data (Liu et al. 2019). Consequently, PLMs are used to transfer the learned knowledge to a new domain or task, and in settings where a model has not seen any example of the required task or interested label during training (known as zero-shot learning) (Yin et al. 2019). These points motivate studying the application of the PLMs for app issue classification.

Additionally, the PLMs that are trained on domain-specific data show significant improvements over general purpose PLMs such as BERT (i.e., PLMs that are trained on general purpose data) for NLP tasks in fields such as science or law (Beltagy et al. 2019; Chalkidis et al. 2020). These domain-specific PLMs leverage unsupervised training on a large domain-specific corpus to compensate for the lack of high-quality, large-scale labeled data in the specified domains. Our problem of interest, the classification of app reviews, aims to extract useful information for software engineers/app developers, which is domain-specific information. As the pre-trained models are trained on general-purpose corpora, app review classification might not benefit from them, as the reviews are short text, many reviews

are noisy and unrelated, and they have a different distribution, which can introduce more complications (Hadi and Fard 2020).

Though a lot of recent studies apply pre-trained neural language models to software engineering tasks (Araujo et al. 2020, 2022; Hadi et al. 2022; Yang et al. 2022; Cao and Fard 2021; Luong et al. 2022; Chen et al. 2022; Karmakar and Robbes 2021; Wan et al. 2022), there is a gap in finding out their applicability for app issue classification. The closest work is the study of Araujo et al. (2020, 2022), where they applied PLMs on one dataset and compared them with classification algorithms. However, whether these models would help issue classification from app reviews over using the *existing approaches*, and whether pre-training them with app reviews helps improve their performance in different scenarios, i.e., multi-class classification, zero-shot classification, multi-task classification, and multi-resource classification, are unknown. So, in this work, we empirically study four PLMs on seven datasets (various sizes and labels) in the mentioned scenarios and explore their performance compared to existing approaches and when the PLMs are trained on domain-specific data.

## 3 Related Works

### 3.1 App Review Classification in Software Engineering

#### 3.1.1 Topic Modeling Based Models

According to Silva et al. (2021), topic modeling has been applied to software engineering research, including app review classification. Among the most frequent topic modeling techniques used to categorize app reviews, the researchers found that Latent Dirichlet Allocation (LDA) (Blei et al. 2003) and LDA-based techniques are the most common. AR-Miner is one of the initial works for mining app reviews proposed by Chen et al. (2014), which uses topic modeling to group the informative reviews and investigate categories of users' discussions. Nayebi et al. (2018) examined how Twitter app reviews can contribute to mobile app development and applied topic modeling and crowdsourcing. Adaptive Online LDA was developed by Gao et al. to classify app reviews based on users' feedback on various versions of mobile apps (Gao et al. 2018). Consequently, Hadi and Fard developed Adaptive Online Biterm Topic Model (AOBTM) (Hadi and Fard 2020) to model topics of app reviews in different categories adaptively, therefore, alleviating the sparsity problems in short-texts, which considered the statistical data for multiple previous time slices.

The topic modeling approach is used in other works (Gao et al. 2018; Guzman and Maalej 2014; Yang et al. 2021; Wardhana et al. 2021; Santiago Walser et al. 2022) to categorize app reviews for detecting emerging issues for developers to update their apps, identifying fine-grained app features in the reviews to extract users' sentiments, prioritizing important user reviews for developers, and performing aspect-based sentiment analysis of app reviews.

#### 3.1.2 Machine Learning Based Approaches

Earlier research has focused on using machine learning approaches for app review filtering. Chen et al. (2014) classified non-informative reviews by training a classifier and categorizing them into informative and non-informative groups. Using a regression model, Fu et al. (2013)

filter out rating-inconsistent reviews that differ in sentiment from their ratings. In Gu and Kim (2015), Gu and Kim identified software features and distinguished between features that are evaluated and those that are requested, using Max-Entropy (Nigam et al. 1999). Other studies used machine learning techniques for app issue classification: (Shah et al. 2018) and (Rustam et al. 2020) use lexical features and multi-text features to use in conjunction with different machine learning models for app review classification tasks. Maalej and Nabil (2015) empirically evaluated different feature extraction techniques to work with machine learning models to classify bug reports and feature requests automatically. Martin et al. (2017) and Sarro et al. (2015) have also explored the breadth of research done on App Store analysis (i.e., feature life cycle, dormant feature identification) for software engineering.

In Guzman et al. (2015); Messaoud et al. (2019); Guo et al. (2019), researchers have used ensemble machine learning techniques to classify app reviews relevant to software evolution, to identify more efficiently and earlier terms in reviews that could be classified into specific topics, and to filter out helpful feedback from users, such as feature requests, bug descriptions, and requirement analysis.

#### 3.1.3 Deep Learning Based Approaches

Using machine learning, Stanik et al. (Stanik et al. 2019) demonstrated that they could still achieve comparable results with Convolutional Neural Networks (CNN) when analyzing user feedback in English and Italian into problem reports, inquiries, and irrelevant. In contrast, Aslam et al. proposed an approach using CNN for app reviews classification with non-textual information to classify app reviews which shows improvements over previous approaches for app review classification (Aslam et al. 2020). In Mekala et al. (2021), a BERT-based sequence classifier was developed and validated to achieve a state-of-the-art average classification accuracy (87%) for feedback analysis. From a large volume of online product reviews, Qiao et al. (2020) proposed to apply a domain-oriented approach to deep learning to discover the most critical users' needs, such as app product new features and bug reports. Henao et al. (2021) investigated the prospect of transfer learning for the classification of app reviews and found that monolingual BERT models outperform existing baseline methods in the classification of English App Reviews.

Other studies (He et al. 2019; Haering et al. 2021; Aralikatte et al. 2018; Wang et al. 2020; Zhao and Zhao 2019; Harkous et al. 2022; Mondal et al. 2022) also applied deep learning techniques to different extents, such as detecting promotion attacks, matching bug reports, detecting adversarial spam, analyzing energy-related reviews, and requirement evolution predictions from app reviews.

### 3.2 Pre-Trained Models

There are two generations of pre-trained models: static and contextual word embeddings (Qiu et al. 2020b). The Global Vectors for Word Representation (Glove) (Pennington et al. 2014), and word2vec (Mikolov et al. 2013) are examples of first-generation static word embeddings. Software Engineering (SE) research has used word embeddings for various tasks, including code retrieval (Van Nguyen et al. 2017), detecting incoherent comments (Cimasa et al. 2019), specifying SE-relevant tweets (Sulistya et al. 2020), and program comprehension with graphs (Allamanis et al. 2018). These kinds of embeddings are context-independent, where each word has only one embedding and cannot change with different contexts. These embeddings

only consider the syntax of the words in a sentence. Therefore, non-contextual embeddings cannot capture semantic information and model polysemous words (Qiu et al. 2020b).

The second-generation pre-trained models learn context-sensitive word representations and can be tailored to perform downstream tasks effectively. LSTM-based (Hochreiter and Schmidhuber 1997) Universal Language Model Fine-tuning (ULMFiT) (Howard and Ruder 2018) and Embeddings from Language Models (ELMo) (Peters et al. 2018) have also been used for different SE-related tasks, such as identifying ambiguous software requirements (Subedi et al. 2021) and sentiment analysis for software engineering (Robbes and Janes 2019). Transformer architectures (Vaswani et al. 2017) have been used to build several pre-trained Transformer-based models (Devlin et al. 2019; Lan et al. 2019; Liu et al. 2019; Yang et al. 2019) that have achieved state-of-the-art performance for a variety of downstream tasks, such as sentiment classification of SE-related texts (Zhang et al. 2020) and identifying error-prone software (Shen et al. 1985). Other studies (Rajpurkar et al. 2016; Reddy et al. 2019; Yang et al. 2018b; Zhang et al. 2021; Ju et al. 2019; Tu et al. 2020) have used PLMs for solving Question-Answering problems. PLMs have also been extensively used for the sentiment analysis task (Bataa and Wu 2019; Sun et al. 2019; Xu et al. 2019; Rietzler et al. 2019; Karimi et al. 2021; Li et al. 2019; Wu et al. 2019; Peters et al. 2017). Previous studies (Liu et al. 2018; Hakala and Pyysalo 2019; Edunov et al. 2019) developed Named Entity Recognition systems with the help of PLMs. Others (Clinchant et al. 2019; Imamura and Sumita 2019; Zhang et al. 2019) studied the applicability of PLMs to improve machine translation tasks. Other works include using PLMs for text summarization (Liu and Lapata 2019; Zhong et al. 2020) and adversarial defense and attack identification (Wallace et al. 2019; Li et al. 2020).

Some recent Software Engineering studies (Araujo et al. 2020, 2022) focused on understanding PLMs for better app review classification. Araujo et al. (2020) used one dataset and four PLMs, including BERT and RoBERTa, to evaluate the classification of the models with four classification algorithms, namely k-Nearest Neighbors, Multinomial Naive Bayes, Support Vector Machines, and Multilayer Perceptron. They experiment with the classification task of the PLMs with and without fine-tuning and find that the bag-of-words, with careful analysis of the text, has comparable results with the PLMs, but PLMs have more advantages. The authors expand their work to opinion mining of app reviews using the same PLMs and classification algorithms (Araujo et al. 2022) and report similar results. PLMs are also explored in other software engineering studies such as API learning (Hadi et al. 2022), API review classification (Yang et al. 2022), mobile app user feedback answer generation (Cao and Fard 2021), finding API-relevant contents (Luong et al. 2022), transfer learning to low resource programming languages (Chen et al. 2022), assessing comprehensive encoding characteristics of source code (Karmakar and Robbes 2021), and structural analysis of source code (Wan et al. 2022).

**Differences between our work and the previous studies** Except for the work of Araujo et al. (2020; 2022) that uses PLMs for app review classification, the current studies develop classification models but do not explore the ability of different models from various perspectives. Even the work of Araujo et al. (2020; 2022) is different from our work in terms of different PLMs used in the experiments, the number of datasets explored with different sizes and labels, and various settings. Other than exploring the models' performances on seven datasets with different labels, our work differentiates from theirs in terms of investigating four scenarios (i.e., the four settings in RQ3), comparing with the current models developed for app review classification (not the general machine learning algorithms used in their study), as well as training Custom PLMs, i.e., PLMs trained on app review dataset. Though the studies

that use PLMs or classify app reviews are numerous, no previous work investigates their capability for app issue classification as extensively as we conduct in this work.

## 4 Study Overview

In this section, we provide an overview of the research design. Figure 1 demonstrates the overall process for RQ1, where we select six widely used datasets, $D_1$–$D_6$ from the literature for app review classification. We also combine $D_1$–$D_6$ and make another dataset, which has multiple labels and is highly imbalanced ($D_7$). We choose four approaches from the literature that we refer to as Prior approaches, namely AR-Miner (Chen et al. 2014), SUR-Miner (Gu and Kim 2015), Ensemble Methods (Guzman et al. 2015), and Deep Learning Model leveraging Non-contextual Word Embedding (DL+WE) (Stanik et al. 2019). These models are selected as they are available and open source, adopted in other studies (Panichella et al. 2015; Villarroel et al. 2016; Di Sorbo et al. 2016; Bavota et al. 2015; Palomba et al. 2015, 2017; Bakiu and Guzman 2017; Martin et al. 2017; Guzman et al. 2016; Ciurumelea et al. 2017; Guzman et al. 2017b; Ali et al. 2017; Scalabrino et al. 2019), and to cover different techniques for app review classification, from mining to deep learning. Four pre-trained neural language models, BERT (Devlin et al. 2019), ALBERT (Lan et al. 2019), RoBERTa (Liu et al. 2019), and XLNet (Yang et al. 2019) are also selected to evaluate their capabilities regarding the research questions in this study. These PLMs are selected as they were previously studied for sentiment analysis in software engineering (Zhang et al. 2020) and are the main architecture for many natural language processing and software engineering models (Feng et al. 2020; Guo et al. 2021). We evaluate all the Prior approaches and the PLMs on each of the seven datasets, $D_1$–$D_7$ separately (RQ1).

From the PLMs, the best performing one from the results of RQ1 is selected and referred to as `PLM-X`, which is used for training on domain-specific (i.e., app reviews) data in RQ2. We note the models trained on domain-specific data as Custom PLMs (`C-PLM-X`). The

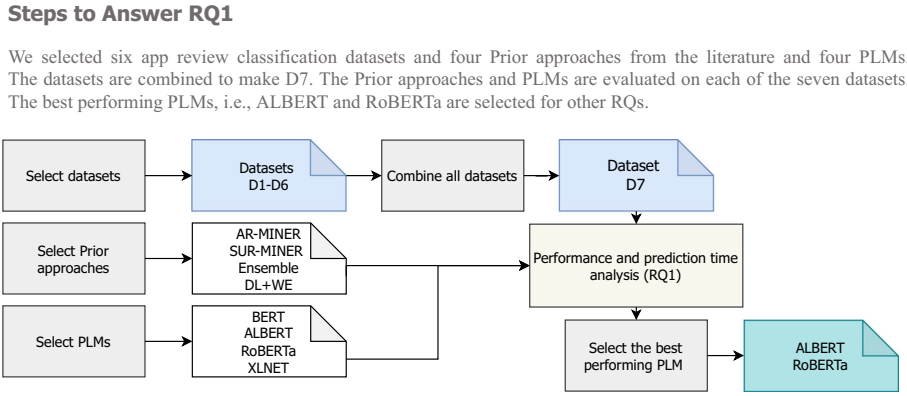


**Fig. 1** Overview of the research steps to answer RQ1. We select six app review classification datasets ($D_1$-$D_6$) from the literature, as well as four Prior approaches, AR-Miner, SUR-Miner, Ensemble, and Deep Learning + Word Embedding (DL+WE), and four PLMs, BERT, ALBERT, RoBERTa, and XLNET. The datasets are combined to make $D_7$. The Prior approaches and the four PLMs are evaluated on each of the seven datasets, and the best-performing PLMs, which, based on the results, are ALBERT and RoBERTa, are selected to be trained with app review dataset, thus making Custom PLMs

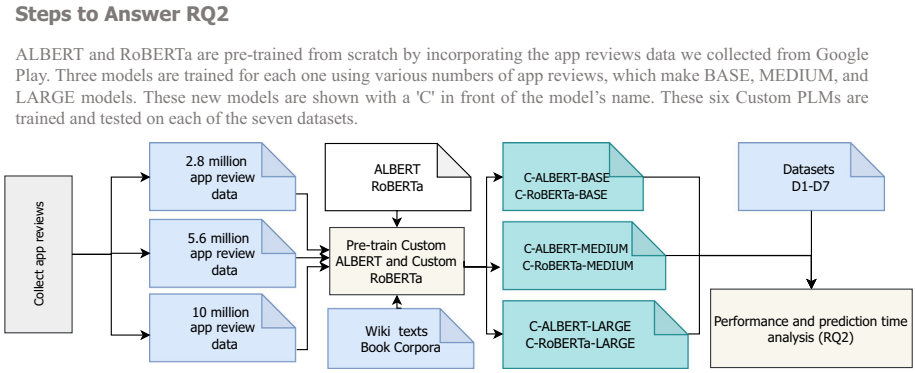


**Fig. 2** Overview of research steps to answer RQ2. ALBERT and RoBERTa are pre-trained from scratch by incorporating the app reviews data we collected from Google Play. Three ALBERTA and three RoBERTa-based models are trained using 2.8 million, 5.6 million, and 10 million app review data, which make BASE, MEDIUM, and LARGE models, respectively. The 'C' in front of the model's name shows this is the Custom PLM, meaning that app review data is used in its pre-training. These six Custom PLMs are trained and tested on each of the seven datasets, $D_1$-$D_7$

`C-PLM-X` is trained using 2.8 million, 5.6 million, and 10 million app reviews integrated with general-domain documents from Wiki-texts and Book corpora. Therefore, we train three models that we refer to as `C-PLM-X-BASE`, `C-PLM-X-MEDIUM`, and `C-PLM-X-LARGE`, for the amount of the app-review data used in their pre-training (see Fig. 2). Note that the '`PLM-X`' in the '`C-PLM-X`' will be replaced with the actual name of the best model, i.e., `C-ALBERT` and `C-RoBERTa` in RQ2 and RQ3, as these are the best-performing PLMs from RQ1, thus, we will have six Custom PLMs: C-ALBERT-BASE, C-ALBERT-MEDIUM, C-ALBERT-LARGE, C-RoBERTa-BASE, C-RoBERTa-MEDIUM, and C-RoBERTa-LARGE. The Custom PLMs are assessed on $D_1$–$D_7$ to answer RQ2.

In RQ3, we evaluate the performance of the PLMs and Custom PLMs in four different settings. For the first part, RQ3-1, the datasets are converted to represent a binary classification, and then the models are assessed in this setting. These steps are shown in Fig. 3. The C-ALBERT and C-RoBERTa models shown in this figure represent all six Custom PLMs to avoid cluttering, C-ALBERT-BASE, C-ALBERT-MEDIUM, C-ALBERT-LARGE, C-RoBERTa-BASE, C-RoBERTa-MEDIUM, and C-RoBERTa-LARGE.

Figure 4 shows the process to answer RQ3-2. The zero-shot classification in this setting assumes that the model is never trained on any labeled data for the task of interest, so the models never see the labels. Therefore, the models and datasets should be prepared according to this setting. Here, the classification is considered a Natural Language Inference (NLI) task (Yin et al. 2019). We prepare the datasets and the models according to the guidelines of Yin et al. (2019), discussing more details in Section 5.5.3.

The steps of RQ3-3 are shown in Fig. 5. The models are evaluated in the multi-task setting. Here, we choose a new app review dataset from Kaggle[1], with the two tasks of i) sentiment classification and ii) identifying the category of the reviews. We evaluate the PLMs and Custom PLMs to classify these datasets.

Finally, Fig. 6 demonstrates RQ3-4, where we evaluate the models in the multi-resource setting, meaning that they are trained on a dataset from one platform and are tested on a

[1] Kaggle is a subsidiary of Google and is an online platform for data scientists and machine learning practitioners: https://www.kaggle.com

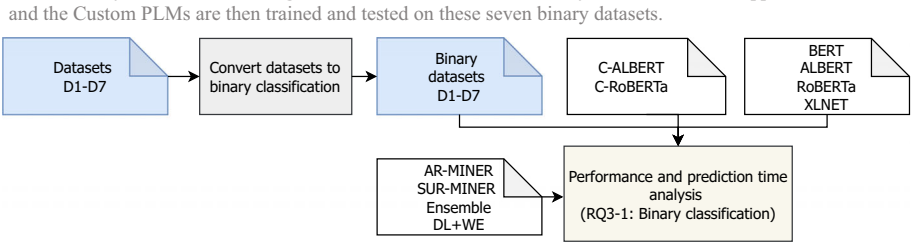


**Fig. 3** Overview of research steps to answer RQ3-1, the binary classification setting. We convert all datasets to binary classes (i.e., only two categories are available in the dataset). The Prior approaches, PLMs, and the Custom PLMs are then trained and tested on these seven binary datasets. The C-ALBERT and C-RoBERTa shown in the figure represent all three BASE, MEDIUM, and LARGE Custom PLMs

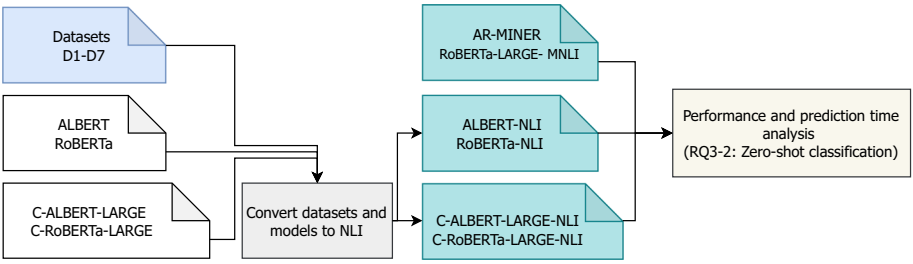


**Fig. 4** Overview of research steps to answer RQ3-2, the zero-shot setting. The datasets, PLMs, and Custom PLMs are converted to the appropriate format, i.e., Natural Language Inference (NLI), to enable testing in zero-shot setting, where the models have not seen *any* of the labels for the task of interest, and are only tested on it. As the LARGE Custom PLMs perform the best, we only assess them along with the RoBERTa-LARGE-NLI on the datasets. These are noted as C-ALBERT-NLI and C-RoBERTa-NLI. From the Prior approaches, only AR-Miner is selected, as this is the only model that can work in this setting

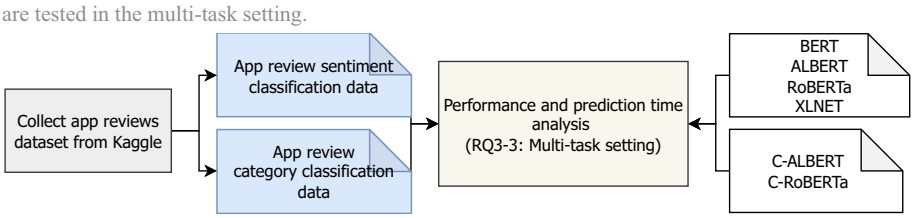


**Fig. 5** Overview of research steps to answer RQ3-3, the multi-task setting. We collect a dataset from Kaggle, which contains app reviews for sentiment classification and app reviews for category classification. We test the PLMs and Custom PLMs in the multi-task setting

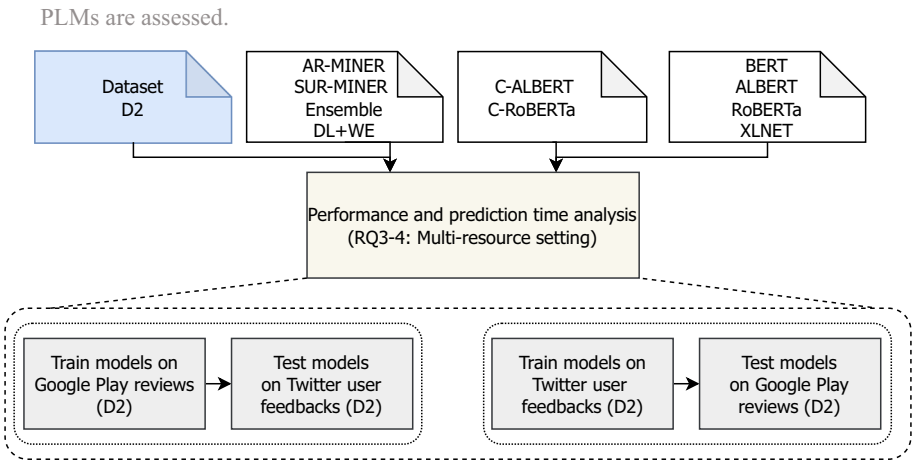


**Fig. 6** Overview of research steps to answer RQ3-4, the multi-resource setting. Here, we use the two parts of $D_2$, which contains both Google Play and Twitter data. The Prior approaches, PLMs, and LARGE Custom PLMs, are assessed for their performance and prediction time

dataset collected from another platform. Dataset $D_2$ contains App Store reviews and Twitter data with the same labels. So, the models will be once trained on the App Store data and tested on the Twitter data, and once trained on Twitter data and then tested on the App Store data. The details of these experiments are described in the next section.

## 5 Experiment Setup

### 5.1 Datasets

App reviews are analyzed automatically for various purposes, and reviews are classified with different labels such as user concerns, feature requests, feature modifications, bug reports, and usability analysis (Hadi and Fard 2020). Our experiments examine the performance of PLMs for app issue classification. Therefore, we select a diverse set of datasets from the literature as described below. These datasets have various sizes, different labels, and are collected from different platforms (i.e., Google Play, Apple App Store, Twitter), and therefore, can represent to what extent the PLMs can be helpful for app issue classification.

**Dataset 1 ($D_1$)** This dataset is procured by Gu and Kim (2015) and contains reviews for 17 popular Android apps from Google Play in the 16 most popular categories, such as games and social. The authors have manually labeled 2,000 reviews for each app (34,000 in total) in five classes according to their predefined rules: Aspect Evaluation (5,937), Praise (8,112), Feature Request (2,323), Bug Report (2,338) and Other (15,290).

**Dataset 2 ($D_2$)** This dataset contains 6,406 app reviews from Google Play and 10,364 tweets (sampled from 5 million tweets written in English), which are labeled manually into three classes: Problem Report, Inquiry, and Irrelevant (Stanik et al. 2019). They included

Problem Report (1,437), Inquiry (1,100), and Irrelevant (3,869) records from Google Play. The number of records in each of these categories from Twitter data is 2,933, 1,405, and 6,026, respectively.

**Dataset 3 ($D_3$)** This dataset is provided by Lu and Liang (2017). Researchers selected two popular Apps, one from Apple App Store (iBooks in the books category) and one from Google Play (WhatsApp in the communication category). They sampled the raw reviews collected from each platform and manually classified 2,000 reviews for each app (4,000 in total). The reviews are classified into six categories: Usability (432), Reliability (587), Portability (119), Performance (121), Feature Request (558), and Other (2,183).

**Dataset 4 ($D_4$)** This dataset was procured by Maalej and Nabil (2015) and contains 2,000 manually labeled reviews from random apps selected from top apps in different categories (1,000 from Apple App Store and 1,000 from Google Play), where half of the apps are paid, and half of them are free apps. The authors ensured 200 reviews for each of the 1, 2, 3, 4, and 5 stars in each of the 1000-review datasets. In addition, they provide 2,400 more reviews that are manually labeled. These reviews are for three selected random Android apps and 3 iOS apps from the top 100 apps (400 reviews for each app). This dataset contains 4,400 labeled reviews in total with four categories: Bug Report (378), Feature Request (299), User Experience (737), and Rating (2721).

**Dataset 5 ($D_5$)** This dataset is published by Guo et al. (2020) and contains 1,500 app reviews, which are manually labeled as User Action (428), App Problem (399), and Neither (673). The reviews are selected from over 5.8 million records for 151 apps from Apple App Store.

**Dataset 6 ($D_6$)** This dataset was procured by Guzman et al. (2015); it contains reviews of 3 apps from the Apple App Store and four apps from Google Play. The apps are popular and from diverse categories. They sampled 260 reviews per app, and five annotators labeled the records, summing to 1,820 reviews that are manually labeled. This dataset includes seven categories, namely: Bug Report (990), Feature Strength (644), Feature Shortcoming (1281), User Request (404), Praise (1703), Complaint (277), and Usage Scenario (593)[2].

*Reasons behind choosing these Datasets ($D_1$ to $D_6$):* We considered these datasets for their diverse characteristics from three perspectives, other than having different categories. *First*, the considered datasets were constructed from various repositories: Apple App Store, Google Play, and Twitter. *Second*, the size of the datasets vary, and they have different number of records for each label. For example, *Dataset 2* is around four times larger than *Dataset 5*. Also, *Dataset 2* contains approximately 2,000 reviews per label, whereas *Dataset 1* has around 400 reviews per label. The size differences can provide insights into the ability of PLMs for different sizes (total training set and the available data for each label). *Third*, except *Dataset 5*, which is a balanced dataset; the other datasets are imbalanced. These differences ensure that we explore the capability of PLMs for different sizes, platforms, and in more realistic settings.

**Merged Dataset ($D_7$)** This dataset will be compiled by merging all six datasets, $D_1$–$D_6$. This *Merged Dataset* has 55,933 app reviews with 16 labels. As some of the labels used in the datasets are the same or have similar definitions, we have grouped some. For grouping the labels, we consulted the class definitions from the studies that published the datasets $D_1$–$D_6$. This dataset would be closer to practical applications where a lot of irrelevant data

[2] Note that adding numbers in all categories will exceed the total number because some reviews belong to multiple groups. We will follow the steps in Guzman et al. (2015) to calculate the evaluation metrics for this dataset.

**Table 1** Description for merged dataset (D7)

| Labels in Merged Dataset | Grouped Labels [Label Name: From Dataset] | # App Reviews in Total |
|---|---|---|
| Performance | Performance: D3 | 121 |
| Portability | Portability: D3 | 119 |
| Usability | Usability: D3 | 432 |
| Reliability | Reliability: D3 | 587 |
| Usage Scenario | Usage Scenario: D6 | 593 |
| Feature Strength | Feature Strength: D6 | 644 |
| User Experience | User Experience: D4 | 737 |
| Feature Shortcoming | Feature Shortcoming: D6 | 1,281 |
| Inquiry | Inquiry: D2 | 1,528 |
| | User Action: D5 | |
| Problem | Problem Report: D2 | 2,113 |
| | App Problem: D5 | |
| | Complaint: D6 | |
| Rating | Rating: D4 (2,721) | 2,721 |
| Bug Report | Bug Report: D1 | 3,706 |
| | Bug Report: D4 | |
| | Bug Report: D6 | |
| Feature Request | Feature Request: D1 | 3,584 |
| | Feature Request: D3 | |
| | Feature Request: D4 | |
| | User request: D6 | |
| Aspect Evaluation | Aspect Evaluation: D1 | 5,937 |
| Praise | Praise: D1 | 9,815 |
| | Praise: D6 | |
| Irrelevant | Other: D1 | 22,015 |
| | Irrelevant: D2 | |
| | Other: D3 | |
| | Neither: D5 | |
| **Total** | | **55,933** |

exists. Multiple classes of informative reviews are of interest to be extracted from this highly imbalanced dataset. The labels in the Merged Dataset, the grouped labels from $D_1$–$D_6$, and the number of app reviews per group are provided in Table 1.

## 5.2 Model Selection

In this section, we discuss the details of the selected Prior approaches and the PLMs, as well as the reasons for choosing each model in our study.

### 5.2.1 Prior Approaches

In the following, we discuss the four widely-used approaches for app review analysis. We have conducted a literature review and reviewed analysis of app review surveys to select these Prior approaches. Another important factor to consider in the recent deep learning-based techniques was the availability of the models.

**AR-Miner** Chen et al. proposed App Review Miner (AR-Miner) (Chen et al. 2014), which extracts valuable information from user reviews. Provided a collection of user reviews, AR-Miner first applies a pre-trained classifier that separates non-informative reviews. Then, AR-Miner applies Latent Dirichlet Allocation (LDA) over the informative reviews to chunk them into different groups for prioritizing them by an efficient ranking model proposed by the authors. AR-Miner has been widely used to mine app issues from user reviews (Panichella et al. 2015; Villarroel et al. 2016; Di Sorbo et al. 2016; Bavota et al. 2015; Palomba et al. 2015). Therefore, it is considered as a Prior approach in our experiments.

*Reason behind choosing AR-Miner:* AR-Miner is an unsupervised approach, and its performance could be evaluated against other models. We mainly choose these techniques to evaluate the PLMs against unsupervised, supervised, and ensemble methods using classical machine learning approaches and, finally, a deep learning technique that is enriched by contextual embedding. AR-Miner is the first framework to accommodate application developers in mining informative topics from a large volume of app reviews. This was the first effective attempt to leverage an unsupervised Topic Modeling technique to extract edifying issues from app reviews. Developers embraced AR-Miner for its ability to filter out non-informative reviews and display key user feedback in an intuitive, concise manner. The effectiveness of AR-Miner was validated by conducting a comprehensive set of experiments on user reviews of four Android apps. Researchers compared the AR-Miner results against real app developers' decisions in different studies (Panichella et al. 2015; Villarroel et al. 2016; Di Sorbo et al. 2016; Bavota et al. 2015; Palomba et al. 2015). They analyzed the advantages of AR-Miner over the manual inspection and other techniques used in a traditional channel (Finkelstein et al. 2014; Johann et al. 2017; Sarro et al. 2015). Based on the empirical results, AR-Miner has proven effective and efficient in extracting informative reviews.

**SUR-Miner** Gu and Kim (2015) proposed Software User Review Miner (SUR-Miner), which is a framework that summarizes users' sentiments, opinions, and emotions toward different aspects of an application. SUR-Miner parses aspect-opinion pairs from reviews by considering their structures. For this purpose, it uses pre-defined sentence templates/patterns[3]. Then, for each review, it combines the sentiment of the sentences with their aspect-opinion pairs. These are used in the final step to summarize the software aspects. SUR-Miner generates reliable summaries and achieved a high F1 score for aspect-opinion identification, sentiment analysis, and app review classification compared to the previous works (Yatani et al. 2011; Guzman and Maalej 2014) by using a simple but effective machine learning algorithm, *Max-Entropy*. It is adopted in many studies (Palomba et al. 2017; Bakiu and Guzman 2017), and therefore we choose it as a Prior approach.

*Reason behind choosing SUR-Miner:* SUR-Miner is the first framework that fully took advantage of user reviews' monotonic structure and semantics; it also defined sentence patterns to extract aspect opinion pairs from app-review. Researchers carefully selected five distinct text features specifically tailored for app reviews and leveraged these features to train a supervised machine learning classifier. Based on the empirical evaluation, we choose this machine learning approach as the terminal classes categorized by the SUR-Miner model was

[3] https://guxd.github.io/srminer/appendix.html

shown to be significantly more accurate and precise than other methods, demonstrating its effectiveness (Yatani et al. 2011; Guzman and Maalej 2014).

**Ensemble Methods** Guzman et al. systematically evaluated different ensemble methods and identified one for classifying user reviews (Guzman et al. 2015). They selected machine learning techniques that were used for text classification and compared the performance of each algorithm individually for app review classification. They also studied the performance of these models when their results were combined using ensemble methods. Ensemble methods are well-known machine learning techniques as they can enhance the prediction performance of single classifiers, maintaining their strengths and reducing their vulnerabilities. In this work, Logistic Regression, Naive Bayes, Support Vector Machines, and Neural Network Classifiers were grouped to vote for the final prediction. This ensemble method outperformed the individual algorithms with statistical significance. It was tested on a large dataset of seven diverse apps, showing that this ensemble method either outperformed or matched its best baseline in all cases. Also, this method influenced further research in the field of app review analysis (Martin et al. 2017; Guzman et al. 2016; Ciurumelea et al. 2017; Guzman et al. 2017b; Ali et al. 2017; Scalabrino et al. 2019) and therefore is used in our study.

*Reason behind choosing Ensemble Methods:* Different studies have attempted to identify the best machine learning approach for app issue classification without spending too much time perfecting custom-engineered features for different issues (Maalej et al. 2016; Wang et al. 2019; Lu and Liang 2017; Wang et al. 2018; Bakiu and Guzman 2017). Guzman et al. (2015) first leveraged different machine learning algorithms and combined their predictions to circumvent the problem of various machine learning models being used to classify issues from app reviews. Their study showed that recall (an evaluation metric) improved significantly after combining the predictions of Logistic Regression and Neural Networks. Software developers and software evolution experts widely adopted this approach to analyze user reviews and prioritize their tasks (Al-Subaihin et al. 2021; Dabrowski et al. 2019).

**Deep Learning Model leveraging Non-contextual Word Embedding** Stanik et al. used a Deep Convolutional Neural Network (CNN) (Stanik et al. 2019) for classifying app reviews. Their model contains an embedding layer, and its weights are initialized with a word embedding model, e.g., word2vec or FastText. This CNN architecture outperforms the shallow Neural Networks classifying app reviews in the determined groups by a small margin. This method used non-contextual word embedding within a deep neural network and performed a transfer of knowledge representation. This approach is one of the first studies leveraging transfer learning and deep learning.

*Reason behind choosing this Deep Learning Approach:* This study is a recent work with the purpose of understanding and evaluating the extent to which deep learning models could be used to categorize user feedback into different predefined categories (Stanik et al. 2019). Based on this study, researchers determined that the domain experts' knowledge incorporation with the traditional machine learning model could achieve comparable results to the deep learning approach due to the substantial performance improvements provided by using simple yet powerful features in the traditional machine learning techniques. Nevertheless, the developers and researchers widely adopted this deep learning approach as the improvement of pre-trained word embeddings brings considerable performance gain to the classifier in other domains (Yang et al. 2018a; Reimers et al. 2019; Ren et al. 2016).

It is worth noting that the number of studies for app review classification is numerous. We chose these four Prior approaches as they are well-cited and widely used in many other studies. More importantly, they use different machine learning and deep learning approaches, representing multiple techniques used in other studies. Additionally, these models are open-sourced

and are publicly available. The best of these four approaches in terms of the performance metrics defined in Section 5.4 will be chosen as the baseline.

### 5.2.2 Pre-Trained Language Models (PLM)

We briefly review our choice of the four PLMs. All these models have Transformer deep learning architecture, which is based on attention mechanism (Yang et al. 2019). As Transformer is currently the leading architecture for PLMs (Liu et al. 2019), we choose the following Transformer-based PLMs: BERT, XLNet, RoBERTa, and ALBERT. The same PLMs are also used for sentiment classification in software engineering (Zhang et al. 2020) and are also the base architecture for other pre-trained models used in NLP and software engineering (Araujo et al. 2022, 2020).

**BERT** Devlin et al. (2019) designed Bidirectional Encoder Representations from Transformers (BERT) to learn contextual word representations from unlabeled texts. Contextual word embeddings designate a word's representation based on its context by capturing applications of words across different contexts. BERT employed a bidirectional encoder to learn the words' contextual representations by optimizing for Masked Language Model (MLM) and Next Sentence Prediction (NSP) tasks. For MLM, 15% of all the tokens are replaced with a masked token (i.e., [MASK]) beforehand, and the model is trained to predict the masked words based on the context provided by the non-masked words. For NSP, the model uses sentence pairs as input to predict whether a pair match is correct or wrong. During training, 50% of the inputs are valid consequent pairs, while the other 50% are randomized non-consequent sentence pairs. Devlin et al. trained two versions: small-sized $\text{BERT}_{BASE}$ and big-sized $\text{BERT}_{LARGE}$. $\text{BERT}_{BASE}$ is a smaller model with 12 layers and 110 million parameters. $\text{BERT}_{LARGE}$ has 24 layers and 340 million parameters. $\text{BERT}_{LARGE}$ is more computationally expensive and consumes more memory compared to $\text{BERT}_{BASE}$. Thus, in this work, we will use $\text{BERT}_{BASE}$.

*Reason behind choosing BERT:* BERT advanced the state-of-the-art for 11 NLP tasks. It achieved an absolute improvement of 7.6% over the previous best score on General Language Understanding Evaluation (GLUE) benchmark[4]; it also achieved 93.2% accuracy on SQuAD 1.1[5], outperforming human performance. We choose BERT because the tasks in these datasets require natural language understanding and include classification tasks.

**XLNet** XLNet (Yang et al. 2019) uses Auto-Regressive language modeling and Auto-Encoding. The model is "generalized" because it captures the bi-directional context using Permutation Language Modeling (PLM). XLNet integrates auto-regressive models and bi-directional context modeling, yet overcoming the disadvantages of BERT. PLM is the idea of capturing bidirectional context on all permutations of terms present in an input sequence. XLNet discards the one-directional linear modeling to maximize the log-likelihood over all permutations of the sequence terms. Each position is expected to learn utilizing contextual information from the entire sequence, thereby capturing the bidirectional context. No [MASK] is needed, and input data need not be corrupted.

In addition, XLNET addresses capturing the dependency between masked positions, which is neglected by BERT. Consider the sentence, "New Delhi is a city." Then, input to BERT to be "[MASK] [MASK] is a city," and the objective of BERT would be predicting "New" given "is a city" and predicting "Delhi" given "is a city". In this objective, there is no dependency between learning "New" and "Delhi." So, BERT can result in a prediction like "New Gotham

[4] https://gluebenchmark.com/

[5] Stanford Question Answering Dataset: https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/

is a city." If we assume that the current permutation is [is, a, city, New, Delhi], BERT would predict the tokens 4 and 5 independent of each other. Whereas XLNet, predicts in the order of the sequence. i.e., first predicts token four and then predicts token 5: it computes the likelihood of "New" given "is a city" plus the likelihood of "Delhi" given "New, is a city".

*Reason behind choosing XLNet:* XLnet outperformed BERT on many NLP tasks; for eight different tasks, XLNet beat BERT by a substantial margin. This model achieved the best results for 18 NLP tasks, including sentiment classification and natural language inference. As XLNET outperforms other models on text classification tasks, we choose it as one of the PLMs in our study.

**RoBERTa** Robustly optimized BERT approach (RoBERTa) outperformed all the state-of-the-art benchmarks upon release (Liu et al. 2019). Liu et al. modified BERT's pre-training steps, which yielded substantially better performance on all the classification tasks. RoBERTa increased the number of mini-batch sizes, data, and training time to train the model. RoBERTa is also trained on a dataset that includes longer sequences than before. The masking pattern in RoBERTa was also modified to be generated spontaneously.

*Reason behind choosing RoBERTa:* RoBERTa outperforms BERT on nine different NLP tasks on the GLUE benchmark; it also equals or exceeds XLNet's model in four out of nine individual tasks. Based on these results, RoBERTa can present a reasonable choice for PLM in our study.

**ALBERT** A Lite BERT (ALBERT) (Lan et al. 2019) applies three parameter reduction techniques: Factorized embedding parameterization, Cross-layer parameter sharing, and Inter-sentence coherence loss. In the first one, researchers separated the hidden layers' size from the input embeddings' size (previously of the same sizes). They projected one-hot vectors to embedding and the hidden space with lower dimensions; it increased the hidden layer size without significantly increasing the vocabulary embeddings' parameter size. Second, all the parameters across all layers are shared. The big-scale ALBERT model has substantially fewer parameters than $\text{BERT}_{LARGE}$. Finally, the NSP task is swapped with Sentence-Order Prediction (SOP) loss which helps ALBERT perform better.

*Reason behind choosing ALBERT:* ALBERT uses parameter reduction techniques which resulted in having 18 times fewer parameters than BERT. Its' training time is 1.7 faster and has negligible inferior performance than the original $\text{BERT}_{LARGE}$ model. The much larger ALBERT architecture, which contains fewer parameters than $\text{BERT}_{LARGE}$, achieved higher F1 scores on the SQuAD 2.0 and the GLUE benchmarks; it also achieved high accuracy on the Reading Comprehension from Examinations (RACE benchmark). Therefore, we chose ALBERT as one of the PLMs in our experiments to evaluate its performance and inference time for app issue classification because ALBERT is shown to be faster than other PLMs without sacrificing much performance.

### 5.3 Cross Validation Training

We train all the Prior approaches and fine-tune all the PLMs on each dataset separately. In the following, we provide general information about training the models using stratified k-fold cross-validation, how PLMs will be trained, and how a classifier is added on top of each model. The detailed process for each research question is provided in Section 4.

To avoid introducing bias to the results due to potential differences in the distribution of the training and test splits, following Stanik et al. (2019), we use k-fold cross validation on each dataset $D_i$ separately, where $D_i$ represents one of the seven datasets in our study. We use k-fold cross-validation as a more rigorous approach than splitting the dataset into train

and test sets once (Zhang et al. 2020). Also, this method has been used previously for app review classification in the software engineering domain (Novielli et al. 2018). In the k-fold cross validation, we split a dataset $D$ into $k$ equal size disjoint parts/folds $D^{(1)}, \ldots, D^{(k)}$. We then build $k$ classifiers $c^{(i)}$, each time using $k-1$ splits as training and one part $D^{(i)}$ as test set. Each data split can be used as a test set once. As a result, we will have $k$ different test set performances. As the datasets in the study are imbalanced, we use an alternate option of cross-validation called stratified k-fold cross-validation (Forman and Scholz 2010; He and Ma 2013). In this method, the only difference is that the distribution of the examples from each class in the original dataset is preserved in each fold $D^{(i)}$. The stratified k-fold cross-validation is commonly used in machine learning practices as it reduces the experimental variance. Therefore, when comparing different methods, it is easier to identify the best method (Forman and Scholz 2010). We consider $k = 5$ as this value has been shown empirically to have test error rate estimates that do not suffer from high bias or high variance (James et al. 2013). The value of $k = 5$ is also used in previous app review classification studies (Stanik et al. 2019). For the evaluation, we use metrics explained in Section 5.4. It is worth mentioning that we apply the stratified 5-fold cross-validation on each of the datasets $D_1$–$D_7$ separately. For example, we split $D_1$ into 5 folds ${D_1}^{(1)}$, ..., ${D_1}^{(5)}$ and build 5 classifiers for $D_1$. We then compute the evaluation metrics for $D_1$. We continue this process for each of the datasets $D_2$–$D_7$ and report the performance of each of them separately.

Note that the number of labels in the datasets $D_1 - D_6$ ranges between [3,7], and all the considered Prior approaches were designed for multi-class classification. Therefore, we can retrain and evaluate them using the stratified 5-fold cross-validation. Among the Prior approaches, only AR-Miner requires some adaptation. AR-Miner has three phases: classification of non-informative reviews, topic modeling of informative reviews, and ranking the groups according to their relevance to developers' applications. As all datasets $D_1$–$D_7$ are labeled, we will directly utilize the second step to group all the reviews (from both training and test set) in a dataset into the corresponding number of labels available. Following the literature (Chen et al. 2014), we will count the training set reviews in a resulting group and use the label with the majority count to annotate the group. This annotation indicates the class that the reviews in the group belong. Then, we will examine the group's test set reviews to compute the evaluation metrics.

### 5.4 Evaluation Metrics

In this section, we describe the evaluation metrics. For all the classifications, we will use three metrics: Precision (P), Recall (R), F1 score (F1), and their micro and macro average values. To answer the time efficiency of the models, we will examine the training time for the Prior approaches and the fine-tuning time for PLMs. We will also report the prediction times and changes in time (increase or decrease of time compared to a baseline).

***Precision (P)*** Precision can be calculated by dividing the number of records their labels are correctly predicted by the total number of predicted observations in that class: $P = \frac{TP}{TP+FP}$. Here, TP refers to the number of records whose label is correctly predicted, and FP refers to the number of records falsely predicted to belong to this class. In multi-class classification, for each group $A$, all the observations that belong to other labels and are falsely predicted as group $A$ are added to compute the FP.

***Recall (R)*** For each group $A$, Recall can be calculated by dividing the number of accurately predicted observations in $A$ by the number of all observations available in the corresponding

class: $R = \frac{TP}{TP+FN}$ Here, FN is the number of observations in class $A$, which are falsely predicted as other labels.

***F1 Score (F1)*** F1 score is the weighted average of Precision and Recall:

$$\mathrm{F1} = \frac{2 \cdot (P \cdot R)}{P + R} \tag{1}$$

We use micro-average and macro-average metrics because we have a multi-class classification, and the datasets are imbalanced. Here, the micro-average calculates the contribution of all records in all classes (therefore, the contribution of the class with the predominant number of records is taken into account), whereas the macro-average is the average of the values for each class (therefore, each class contributes equally in the final value). The micro- and macro-averaged precision (P) are computed as:

$$P_{micro} = \frac{\sum_{j=1}^{m} TP_j}{\sum_{j=1}^{m} TP_j + \sum_{j=1}^{m} FP_j} \tag{2}$$

$$P_{macro} = \frac{\sum_{j=1}^{m} P_j}{m} \tag{3}$$

$TP_j$ and $FP_j$ are the number of true positive and false positive predictions for the $j$-th class, respectively. $P_j$ is the precision for class $j$ and $m$ is the number of classes. Similarly, the micro- and macro-scores of Recall (R) and F1 score will be calculated and denoted as $\mathrm{R}_{micro}$, $\mathrm{R}_{macro}$, $\mathrm{F1}_{micro}$, and $\mathrm{F1}_{macro}$.

As we are using stratified 5-fold cross-validation, for each dataset $D$, we will report averages of these metrics obtained from each of the k classifiers:

$$F1^{\mathrm{avg}}{}_{\mathrm{micro}} = 1/k \sum_{n=1}^{k} F1_{\mathrm{micro}}{}^{(\mathrm{i})} \tag{4}$$

$$F1^{\mathrm{avg}}{}_{\mathrm{macro}} = 1/k \sum_{n=1}^{k} F1_{\mathrm{macro}}{}^{(\mathrm{i})} \tag{5}$$

The $F1_{\mathrm{micro}}{}^{(\mathrm{i})}$ and $F1_{\mathrm{macro}}{}^{(\mathrm{i})}$ refer to the test performance of classifier $c^{(\mathrm{i})}$ on held out test set $D^{(\mathrm{i})}$. Similarly, the $\mathrm{P}^{\mathrm{avg}}{}_{micro}$, $\mathrm{P}^{\mathrm{avg}}{}_{macro}$, $\mathrm{R}^{\mathrm{avg}}{}_{micro}$, and $\mathrm{R}^{\mathrm{avg}}{}_{macro}$ will be computed.

Following previous works (Zhang et al. 2020), *if a model has higher values for both $F1^{avg}{}_{micro}$ and $F1^{avg}{}_{macro}$, we consider it to be better than other models.*

***Time*** To compare the time efficiency, we will report the training time for other approaches and fine-tuning time for PLMs and the prediction time of all the approaches. Prediction time is the time the model requires to process the test data and predict labels. The time will be measured in seconds and reported for each dataset. *A model is considered to be more time efficient than another if the prediction time is less;* as training a model or fine-tuning a PLM is a one-time process and needs not to be repeated.

***Time Change in Percentage*** We employ all four Prior approaches on all datasets $D_1$–$D_7$ and choose their best as our *baseline* approach. In addition to the Time mentioned above, we will also report the increase and decrease in percentage for the studied PLMs concerning the considered baseline for the measured Time. The reason for reporting this change in percentage is that time duration depends on hardware configuration, and percentage change provides readers with a better understanding of the change in time.

## 5.5 Experiment Design

### 5.5.1 RQ1 Experiment Design

For RQ1, we identify the accuracy and efficiency of PLMs compared to Prior approaches. For this RQ, all Prior approaches and PLMs will be trained and tested on all datasets separately, as explained in Section 5.3. We only consider the reviews (not tweets) from *Dataset 2* for RQ1. All the evaluation metrics described in Section 5.4 will be reported for each dataset and approach. We highlight the best model for each dataset with the highest $F1^{avg}{}_{micro}$ and $F1^{avg}{}_{macro}$. The lowest training/fine-tuning and prediction times will also be highlighted for each dataset, along with the time change in percentages. We denote the best performing PLM among others as `PLM-X` to pursue experiments in other research questions. If no PLM achieves the best results in terms of both micro- and macro-F1 score, the PLM that achieves the highest $F1^{avg}{}_{micro}$-score (denoted as `PLM-XM1`) and the PLM with the highest $F1^{avg}{}_{macro}$-score (denoted as `PLM-XM2`) will be selected to be studied in RQ2.

### 5.5.2 RQ2 Experiment Design

In RQ2, we are interested in evaluating the performance of PLMs when they are pre-trained on domain-specific corpus rather than non-domain-specific corpora. In RQ2, we will use `PLM-X` (or `PLM-X1` and `PLM-X2`) from RQ1 and all seven datasets. We can either pre-train `PLM-X` using just the domain-specific app reviews or combine them with the public domain documents to pre-train `PLM-X` from scratch. In the first approach, `PLM-X` with the same architecture will be trained from scratch on app review sentences. However, pre-training `PLM-X` on a small number of domain-specific documents has its disadvantages; the model may overfit our dataset, and it can result in performance degradation in downstream tasks. By pre-training PLMs using both general and domain-specific datasets, we can avoid this problem. We use the approach of Wada et al. (2020), where we pre-train the `PLM-X` simultaneously with both general and domain-specific documents. Following the literature (Wada et al. 2020), we double the frequency of pre-training domain-specific documents during the optimization of the Masked Language Modeling (MLM) task, which is the task the ALBERT and RoBERTa models are trained.

To accomplish this, we must increase the frequency of pre-training for MLM over app reviews. The objective can be achieved by improving the vocabulary representation of app reviews. Therefore, we use simultaneous pre-training after the up-sampling introduced by Wada et al. (2020). We create pre-training instances from a set of corpora with different sizes to pre-train a PLM, as described in Fig. 7. Upsampling refers to the copying and pasting of the app reviews portion of the corpus until the predetermined ratio of app reviews and general pre-training data is achieved; also, we have uniformly mixed the book documents, wiki documents, and app reviews consecutively to form the pre-training corpus. With this up-sampling technique, more instances from the small app review corpus are used for MLM.

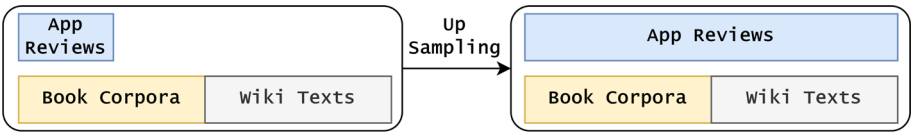


**Fig. 7** We up-sample the app reviews data and increase their frequency for the pre-training of Custom PLMs

We will pre-train three different PLM-Xs, where we will respectively integrate 2.8 million, 5.6 million, and 10 million app reviews that we have collected from Google Play, with the public domain documents from Wiki-texts[6]. Our collected dataset has app reviews for more than 2000 apps from different categories. The dataset includes app_name, app_category, review, rating, reply_text, and date. This will give us three versions of a domain-specific pre-trained model, which we refer to as *Customized PLM-X* (C-PLM-X). We denote the three sizes of the C-PLM-X as C-PLM-X-BASE, C-PLM-X-MEDIUM, and C-PLM-X-LARGE, respective to the number of the app review sentences integrated for pre-training. Finally, we will fine-tune all the C-PLM-Xs on the seven datasets according to the steps, and evaluation metrics explained in Sections 5.3 and 5.4. The best C-PLM-X will be used in RQ3.

### 5.5.3 RQ3 Experiment Design

We have used both the readily available PLMs and our C-PLM-X in different settings for evaluating RQ3. We are considering four different settings to determine the PLMs' capacity regarding app review analysis.

**Binary vs. Multi-Class Setting** Previous studies have shown that the pre-trained Transformer based models deviate from their original performance when the classification task involves multiple classes instead of binary classes (Adhikari et al. 2019; Chang et al. 2019). Studies on app review classification also show that binary classification yields better results (Jha and Mahmoud 2019; Maalej et al. 2016). We are, therefore, interested in evaluating the performance of the PLMs in both binary and multi-class settings, especially investigating the performance of C-PLM-X. We will investigate the accuracy and time efficiency of PLMs for binary classification compared to the original multi-class settings. We convert all seven datasets into binary-class datasets by randomly choosing one class (i.e., label) for each dataset and switching all the other labels in that dataset into "Other", considering the randomly selected class as positive and the other classes as negative class. Then, we will use these modified datasets to regenerate the PLM and C-PLM results for the binary classification task on each dataset. The four PLMs and C-PLM-X will be used to record the evaluation metrics. All other steps will remain similar to RQ1.

**Zero-Shot Classification Setting** The Zero-Shot Learning (ZSL) that we will investigate in our study is the recent approach in which we evaluate the models on fully-unseen labels. In this setting, we assume that the system is never trained on any labeled data for the task we are interested in, which is different from some ZSL settings in which the model partially sees labels. The setting we choose here is more realistic and can provide insights about the potential benefits of PLMs in practice, where the developers might want to use the same model to derive new unseen aspects of the dataset. Using PLMs for this kind of zero-shot classification requires different training.

For ZSL, we will explore the method proposed by Yin et al. (2019) in which classification is considered a Natural Language Inference (NLI) task. This approach determines the compatibility of two distinct sequences by embedding both sequences and labels into the same space. In NLI, a pair of sentences are considered: "premise" and "hypothesis," and the task is to predict whether the "hypothesis" is an entailment of "premise" or not (contradiction). We follow the steps in Yin et al. (2019) to prepare the datasets and models and set up our study for ZSL. We will use the reviews as "premise", and candidate labels as "hypothesis." For Example, for the app review "Please add a back button in the armory page." and its label "Feature Request," we use the review as a *premise* and the label as a *hypothesis*. So, the NLI

[6] The Wiki-texts dataset is open sourced at https://dumps.wikimedia.org/.

model can predict whether the hypothesis is an entailment of the premise. This prediction will be compared to the actual labels for the reviews to calculate the evaluation metrics. If the NLI model correctly predicts the *hypothesis* as an entailment of the *premise*, it is recorded as a True Positive. On the other hand, if the NLI model predicts the *hypothesis* as contradictory or neutral to the *premise*, it is recorded as a False Negative. If the NLI model predicts the *hypothesis* as contradiction or neutral, where the *hypothesis* is actually an entailment, we refer to this as False Positive (FP). On the other hand, if the NLI model predicts the *hypothesis* as contradiction or neutral, where the *hypothesis* is actually contradiction or neutral, it is regarded as True Negative (TN).

We fine-tune `C-PLM-X` on NLI dataset[7] and test it on the seven datasets to determine how well PLMs can classify issues from app-reviews without being trained on them. Note that the NLI data and app reviews are both in natural language, but the domains are different, and NLI data does not contain app reviews. So, the models are fine-tuned on a dataset that does not contain app reviews but is only tested on them. In addition, we use `RoBERTa-large-nli`[8], which is a readily available fine-tuned model for zero-shot classification [9]. This will provide insights about using a custom PLM trained on app reviews and the PLM trained on general-purpose corpora in a zero-shot setting.

**Multi-Task Setting** Research has confirmed that when a new classification task (e.g., sentiment classification) is introduced, a new model should be trained, and the models for another classification task (e.g., app issue classifiers) cannot be applied for this new task (Stanik et al. 2019). In addition, the same model or classification technique can have different performances on various analysis tasks. For example, Hemmatian and Sohrabi found Naive Bayes (a probabilistic classifier) to be performing better than Decision Tree (a non-probabilistic classifier) for sentiment classification tasks (Hemmatian and Sohrabi 2019); on the other hand, Maalej et al. reported that using Bag-of-Word technique and Decision Tree outperformed Naive Bayes for app issue classification task (Maalej et al. 2016). Therefore, we are interested in evaluating the performance of PLMs in both app issue classification and sentiment analysis task settings. In sentiment analysis, the task is to classify the sentiment of a given review into positive, neutral, or negative polarities (Zhang et al. 2020), which is different from the app issue classification. In this setting, we will evaluate the `PLMs` and `C-PLM-X` on the new task of sentiment classification for app reviews. We fine-tune these models on the sentiment classification dataset and test them for sentiment classification. This setup will help evaluate whether the models trained for app issue classification can be applied to another task, thus providing insights into the multi-task ability of the PLMs. For the sentiment classification, we will use another dataset[10] that consists of 37,185 app reviews with three sentiment polarity labels: *positive* (13,758), *neutral* (9,993), and *negative* (13,434).

We have also gone beyond sentiment classification and implemented the PLMs for category classification. The same Kaggle datasets also contain category information for 20,955 app reviews. The number of user reviews in these categories are: *Family* (18.9%), *Games* (50.3%), *Tools* (12.5%), *Medical* (10.7%), and *Business* (7.6%). As it is an imbalanced dataset, we use stratified K-fold cross-validation to maintain uniformity with other experiments. The experimental settings and evaluation metrics we will use are similar to RQ1. The

[7] https://cims.nyu.edu/~sbowman/multinli/

[8] https://huggingface.co/joeddav/xlm-roberta-large-xnli

[9] We only use the readily available PLMs for this study and avoid fine-tuning other PLMs on Multi-NLI data in this setting, due to high computational expenses.

[10] https://www.kaggle.com/lava18/google-play-store-apps

results will determine how these models perform for different classification tasks in the same domain (app reviews).

**Multi-Resource Setting** The data distribution from multiple resources varies, and models built for App Stores are unsuitable for classifying user feedback from another resource, such as Twitter (Guzman et al. 2017a). In this setting, we will study the accuracy and time efficiency of the PLMs for the classification of data collected from multiple resources. For this experiment, *Dataset 2* will be used as it contains App Store reviews and Twitter data with the same labels. First, we will fine-tune the `PLM-X` and `C-PLM-X` on App Store reviews of *Dataset 2* and report the evaluation metrics (training steps are similar to RQ1) when tested on Twitter data. In addition, we will fine-tune the `PLM-X` and `C-PLM-X` on the Twitter part of *Dataset 2* and report the results when tested on App Store reviews. This way, we will gain insights into the PLMs' capacity to classify app reviews collected from different resources on which the PLMs were not fine-tuned. The evaluation metrics used here are explained in Section 5.4. To compare, we also test the Prior approaches for this setting. From the Prior approaches, AR-Miner does not require training. For the other approaches, we train them on the App Store reviews of *Dataset 2* and test them on Tweets in *Dataset 2*.

## 5.6 Implementation

We discuss the implementations of both Prior and PLM approaches in this section.

### 5.6.1 Prior Approaches

**AR-Miner** uses LDA (Blei et al. 2003) topic modeling algorithm to categorize text document corpus into a predefined number of related and coherent topics. For app review classification, we hid the labels of the considered datasets and presented the app reviews as a text corpus, where each review represented different documents. If a dataset has $n$ classes, we will extract $n$ number of topics from the app review corpus; the documents will be separated into $n$ number of segments. After extracting topics, we looked into the associated labels for each review for assigning each document cluster (topic) to a relevant class/label. The relevance was measured by the prevalence of a class or label in a topic after extraction. Suppose the highest number of app reviews belong to label/class $x$ in a topic (document cluster); we would understand that AR-Miner has labeled all documents in that cluster with class $x$, where the app reviews that genuinely belonged to class/label $x$ are regarded as the True Positives, and the rest of the reviews are regarded as False Positive.

**SUR-Miner** used *Max-Entropy* model to classify app reviews. This classifier model considers all probability distributions empirically consistent with the training data and chooses the distribution with the highest entropy. Empirically consistent probability distribution with training data is one in which an estimated class occurrence frequency and a feature value are equal to the actual frequency in the training data. Following the literature (Gu and Kim 2015), we have implemented Generalized Iterative Scaling (GIS) as our scaling method and performed Early Stopping to avoid over-fitting and to stop the iterative solver from taking a long time to optimize over a large number of feature weights. As mentioned by Gu and Kim (2015), we have also adopted 2-4 Grams (Character N-Gram) for the classification task.

**Ensemble Approach** adopted an ensemble of Logistic Regression classifier and shallow Neural Network classifier and applied the majority voting scheme to combine the output of the classification approaches by following the literature (Guzman et al. 2015). We have concatenated TF-IDF vector representations of app reviews, the number of words in the

**Table 2** Details of PLMs used in our study

| Architecture | Used Model | Parameters | Layers | Hidden | Heads |
|---|---|---|---|---|---|
| BERT | bert-base-cased | 110M | 12 | 768 | 12 |
| RoBERTa | roberta-base | 125M | 12 | 768 | 12 |
| XLNet | xlnet-base-cased | 110M | 12 | 768 | 12 |
| ALBERT | albert-base-v1 | 11M | 12 | 768 | 12 |

[1] BERT: https://huggingface.co/bert-base-cased
[2] RoBERT: https://huggingface.co/roberta-base
[3] XLNET: https://huggingface.co/xlnet-base-cased
[4] ALBERT: https://huggingface.co/albert-base-v1

review, the number of characters in the review, the number of lowercase characters, number of uppercase characters to use as features for our ensemble approach. We have used the Adam optimizer and binary cross entropy as our loss function in the shallow neural network.

**Deep Learning Approach With Word Embeddings** followed Stanik et al. (2019). Text inputs must have a fixed size for the neural network's input layer. This input size has been fixed at 400 words, a measure we found suitable for all the app reviews in all of the datasets we studied, and we identified no app reviews that exceeded 400 words. In addition to the input layer, our network consists of an embedding layer (pre-filled with a fast-text model), a 1D convolution layer, a 1D global max-pooling layer, a dense layer, and a final output layer with a softmax activation. Tanh activation was used for the previous layers. The embedding layer weights were frozen during training, leaving the trainable parameters.

### 5.6.2 PLMs

For evaluating the PLMs, we use the training set for each dataset to fine-tune the PLMs and then evaluate them on the test set. We obtain all four PLMs pre-trained on public (general) domain corpora from Hugging Face library[11]. Their base versions are used for BERT, XLNET, RoBERTa, and ALBERT. Table 2 includes the names of the models used from the Huggingface Transformers (Wolf et al. 2019) and the default configurations. The mentioned PLMs provide contextual embeddings for our purpose. To build a classifier, a feed-forward dense layer and softmax activation function are added to each model. We follow Zhang et al. (2020) and Devlin et al. (2019) to set the hyperparameter values. In our work, we set the batch size to 16 and Adam's learning rate to $2e-5$. The models are trained for four epochs, and *AdamW* optimizer is used for all models.

We execute all the experiments on a Linux machine with Intel 2.21 GHz CPU and 16GB memory. For training PLMs from scratch, we use $2 \times$ NVIDIA Tesla V100 32GB to enhance the parallelization performance.

## 6 Results

In this section, we present the results of our experiments for each of the research questions. Note that we have computed all the scores (micro- and macro-scores of P, R, F1) and the training and prediction times for all of the models for each dataset. Nevertheless, due to the

[11] https://huggingface.co/

large number of experiments, we bring all tables in the appendix. In the following, we only discuss the F1 scores and the prediction time. The full results are shown in Tables 3 and 4 for RQ1, Tables 5 and 6 for RQ2, Tables 7 and 8 for RQ3.1, Table 9 for RQ3.2, Table 10 for RQ3.3, and Table 12 for RQ3.4 in the appendices.

### 6.1 RQ1: Accuracy and Efficiency of PLMs Compared to Prior Approaches

Figure 8 shows the micro-F1 scores for Prior approaches and PLMs, where the performances of all models are shown for each dataset. Figure 9 represents another perspective of the micro-F1 scores where the scores are separated for each dataset. Among the Prior approaches, *DL+WE* is the best-performing model, followed by *Ensemble* approach. Only for $D_5$, which has the lowest number of app reviews, the *Ensemble* method has slightly better scores than *DL+WE*. We conjecture that this low number of reviews favored the *Ensemble* model to enhance the prediction performance of participating classifiers. For the cumulative dataset

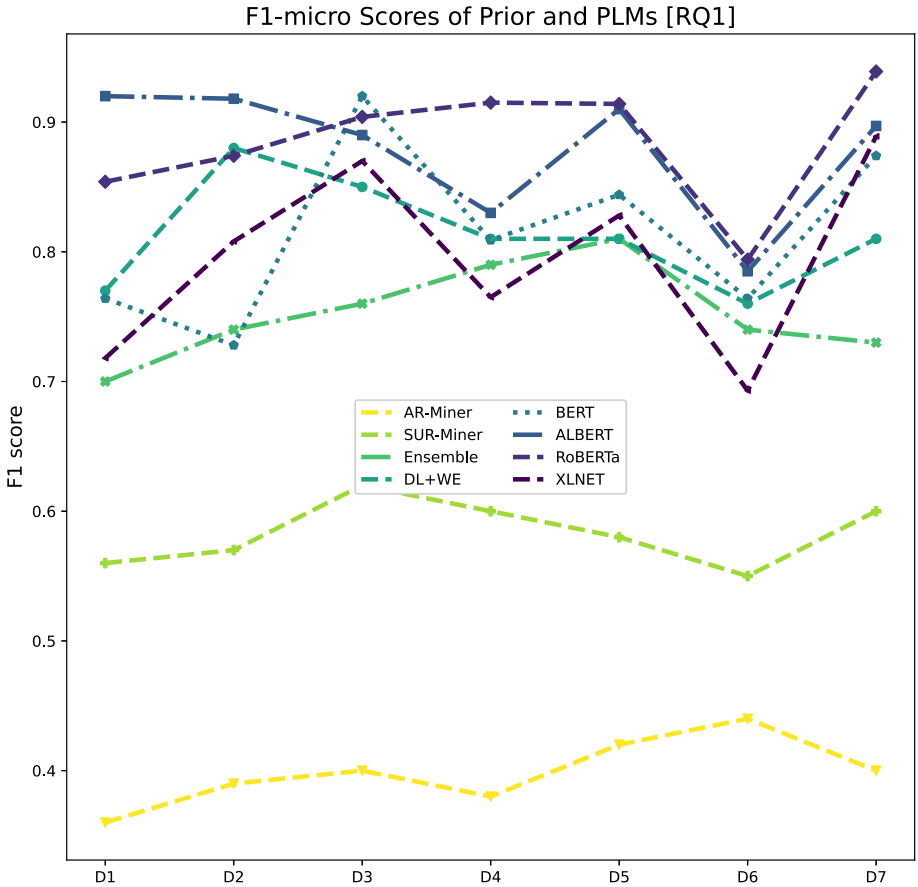


**Fig. 8** F1-micro scores of the Prior and PLMs for app issue classification in datasets $D_1$–$D_7$ (RQ1). Each model is shown in color. ALBERT and RoBERTa perform better than other models. From the Prior approaches, Ensemble and DL+WE have higher scores

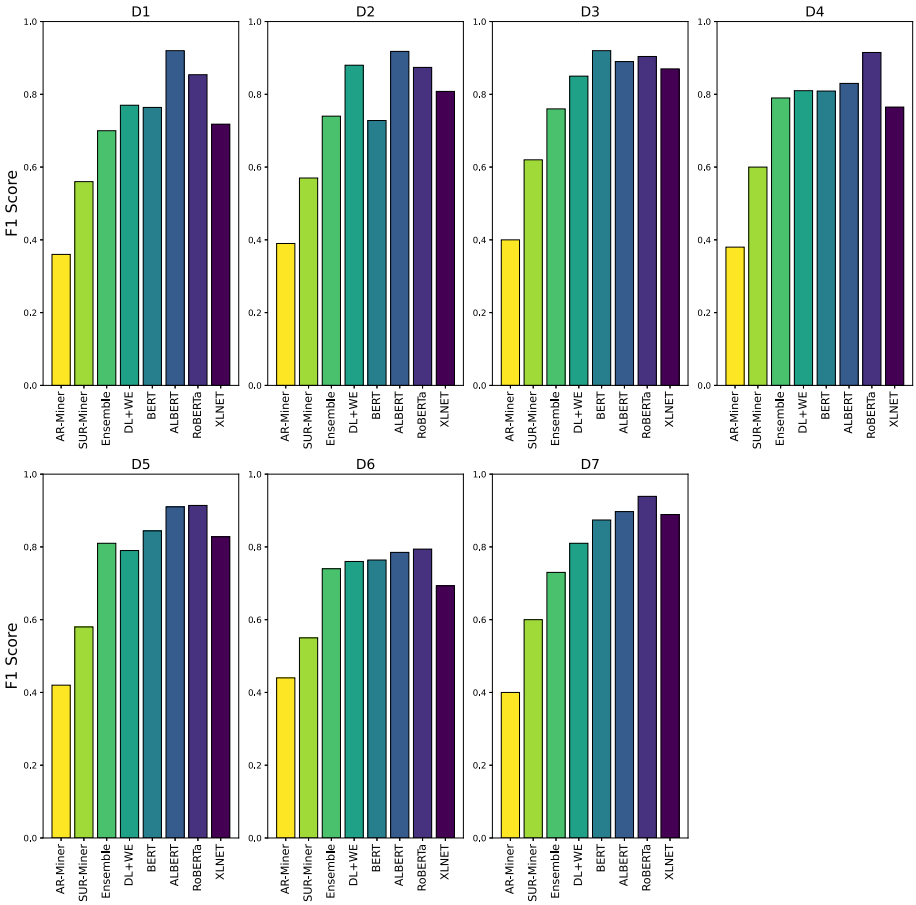


**Fig. 9** F1-micro scores of the Prior and PLMs for app issue classification in datasets $D_1$–$D_7$ (RQ1), separated for each dataset. Each model is shown in color. ALBERT and RoBERTa models perform better than other models. From the Prior approaches, Ensemble and DL+WE have higher scores

(*Dataset 7*), *DL+WE* improves the results of *Ensemble Method*, *SUR-Miner* and *AR-Miner* by ∼8%, ∼20%, and ∼40%, respectively. So, we selected *DL+WE* as our baseline from Prior approaches.

Among the PLMs, on most datasets, *ALBERT* and *RoBERTa* achieve the highest F1-micro and F1-macro scores. The F1-micro results of these PLMs are higher than the best Prior Approach and other PLMs for all datasets, except that BERT outperforms other models for $D_3$. The *Ensemble* and *DL+WE* approaches can match the performance of some worse-performing PLMs in the respective dataset, such as *DL+WE* outperformed *BERT* on *Dataset 2*. The other Prior approaches (i.e., AR-Miner and SUR-Miner) have significantly lower scores than the PLMs for all Datasets. The performance of *XLNET* varies compared to Prior approaches. For some datasets such as $D_1$, $D_4$, and $D_5$, it has comparable or lower micro-F1 compared to *Ensemble* and *DL+WE*; but for $D_7$ it has higher scores. *RoBERTa* performs

best on all metrics for *Datasets 4 and 6*, and yields best micro-average scores for $D_5$. These three datasets have the least number of app reviews. On the other hand, *ALBERT* performs best on Datasets $D_1$ and $D_2$, the two datasets with the highest number of app reviews. For the cumulative dataset (Dataset 7), *RoBERTa* outperforms the other PLMs on micro-scores, *XLNET* has the highest macro-scores, and the other PLMs have relatively close results. For $D_3$, the established order of PLM's performance persists (*RoBERTa* >*ALBERT* >*XLNet*), except for the performance yielded by *BERT*. *BERT* outperformed the other three PLMs by a small margin for this dataset (∼1.5% to ∼4.5%).

The change in the models' performances with different datasets can be related to the size of the data, the number of app reviews in each class, the label of the class (e.g., Portability vs. Complaint), and the number of classes presented in each dataset. These differences can affect the classification scores. For example, the number of reviews per class can be a characteristic affecting the scores here. Dataset $D_2$ contains 6,406 app reviews in three classes of Problem Report, Inquiry, and Irrelevant, whereas dataset $D_3$ has 4,000 reviews in six groups Usability, Reliability, Portability, Performance, Feature Request, and Other. Looking at Tables 3 and 4 in the Appendix, the best macro averages of dataset $D_2$ are higher than the best macro averages of $D_3$. As the macro-averages consider the contribution of each class in the final value equally, this shows that the prediction scores for some classes in the third dataset are lower (thus contributing to the final macro-scores).

By exploring $D_3$ in more detail, we observe that the 'Other' class has more than 2,000 records, meaning that the models learn about five classes from a small portion of reviews. Additionally, when we further analyzed the confusion matrices of ALBERT and RoBERTa for $D_3$, we found that the main reason for dropping their performance in this dataset is that they predict the records of the 'Other' class wrong. However, their prediction for the other five classes is mostly correct. So, the False Negative predictions are higher, which is also confirmed by having higher precision than recall for these models in $D_3$. This suggests that not only the number of reviews per category is important, but it seems the classes (labels) of interest in each dataset could contribute to the performance of the models.

To show the differences among the number of records for each label in datasets $D_1$–$D_7$, we plot the number of the predicted records for each class alongside the Ground Truth in Figs. 20, 21, 22, 23, 24, 25 and 26, shown in Section Appendix. Consider *Dataset 1* which has a large number of records (15,290) labeled as 'Other' out of 34,000 records. The rest of this dataset is labeled by four other classes. On the other hand, $D_6$ has 1,820 reviews that are classified into seven categories, Bug Report, Feature Strength, Feature Shortcoming, User Request, Praise, Complaint, and Usage Scenario. Therefore, in $D_1$, the number of available reviews per label is much higher than the number of reviews for each label in $D_3$ and $D_6$. However, the macro-scores for datasets $D_3$ and $D_6$ are higher or in par with $D_1$. These observations show that different factors contribute to the models' performance for datasets.

As explained, the best performing PLM changes with different datasets, which can be related to the characteristics of the datasets we discussed above, as well as the differences among the PLMs, including the size of the dataset used in their pre-training and the tasks they are trained on. For example, RoBERTa is pre-trained on a larger dataset, and ALBERT uses Sentence Order Prediction, which might help classify different categories. As both *RoBERTa* and *ALBERT* have the highest or second-highest scores in maximum number of datasets, we choose these two PLM models as the best-performing ones to pursue with the following research questions. These two PLMs will be our PLM-X models and are chosen for training

from scratch to develop `C-PLM-X (Custom PLMs)`. We refer to these Custom PLMs as C-ALBERT and C-RoBERTa.

Figures 16 and 17 (see Section Appendix) illustrate the prediction time for Prior and PLM approaches in seconds, the former showing the times per model and the latter demonstrating the prediction time per dataset. As *AR-Miner* uses topic modeling to predict the labels/classes; so, the prediction time is not reported for *AR-Miner*. From Fig. 16, we can see that both Prior and PLM approaches follow a similar trend, and their prediction times are related to the size of the dataset, i.e., the larger the size of the dataset, the longer the prediction time is. Among the Prior approaches, our previously selected baseline *DL+WE* yielded the least prediction time on three datasets ($D_1$, $D_4$, and $D_7$), whereas *SUR-Miner* method's prediction time is the least for rest of the datasets ($D_2$, $D_3$, $D_5$, and $D_6$). Among the PLMs, *ALBERT* consistently yields the least prediction time for all the datasets. This can be related to the fact that ALBERT is the light BERT, so, in most results, it has the lowest prediction times. On the other hand, the prediction time for *XLNet* is the highest for all datasets.

In the last column of Table 4 (*Time-Diff (%)*), we have included the percentage change of the prediction times of PLMs with respect to the prediction time of the Prior approach that has the lowest time. For example, for $D_1$, the Time-Diff (%) for PLMs is calculated based on the prediction time of *DL+WE*; and for $D_2$ it is calculated based on the prediction time of *SUR-Miner*. For all datasets except $D_7$, the prediction time of PLMs compared to the best Prior approach increases by 0.49% to ∼500%. For $D_7$, *BERT* and *ALBERT* predicted the classes ∼ 3% and ∼ 22%, respectively, faster than the fastest Prior approach (*DL+WE*). For $D_7$, RoBERTa increases the time by 0.15%, and XLNET increases the time by ∼89%.

**Findings of RQ1**

PLMs are generally superior to Prior approaches when predicting classes with higher scores across all datasets. The two candidates for the best PLMs we choose are ALBERT and RoBERTa, as they invariably perform well across all dataset sizes. Based on the time taken for prediction, DL+WE took the shortest amount of time among all Prior and PLM approaches, and ALBERT took the shortest amount of time among the PLM approaches.

### 6.2 RQ2: Domain-Specific PLMs vs. General PLMs

Figure 10 shows the performance of *C-ALBERT* and *C-RoBERTa* after they are pre-trained on domain-specific app-reviews along with general domain corpora. In `C-PLM-X-BASE`, we have incorporated 2.8 million app reviews during pre-training. For `C-PLM-X-MEDIUM` and `C-PLM-X-LARGE`, we have increased the number of app reviews during pre-training to 5.6 million and 10 million, respectively. The 'B,' 'M,' and 'L' in the model names are abbreviations of 'BASE,' 'MEDIUM,' and 'LARGE' in the model names.

Another perspective of the same data is shown in Fig. 11, where the scores are separately shown for each dataset. From the result, we can see that all the Custom PLMs performed better in all datasets than their corresponding out-of-the-box PLMs. Increasing the number of app reviews in pre-training corpora helps boost the performance of Custom PLMs.

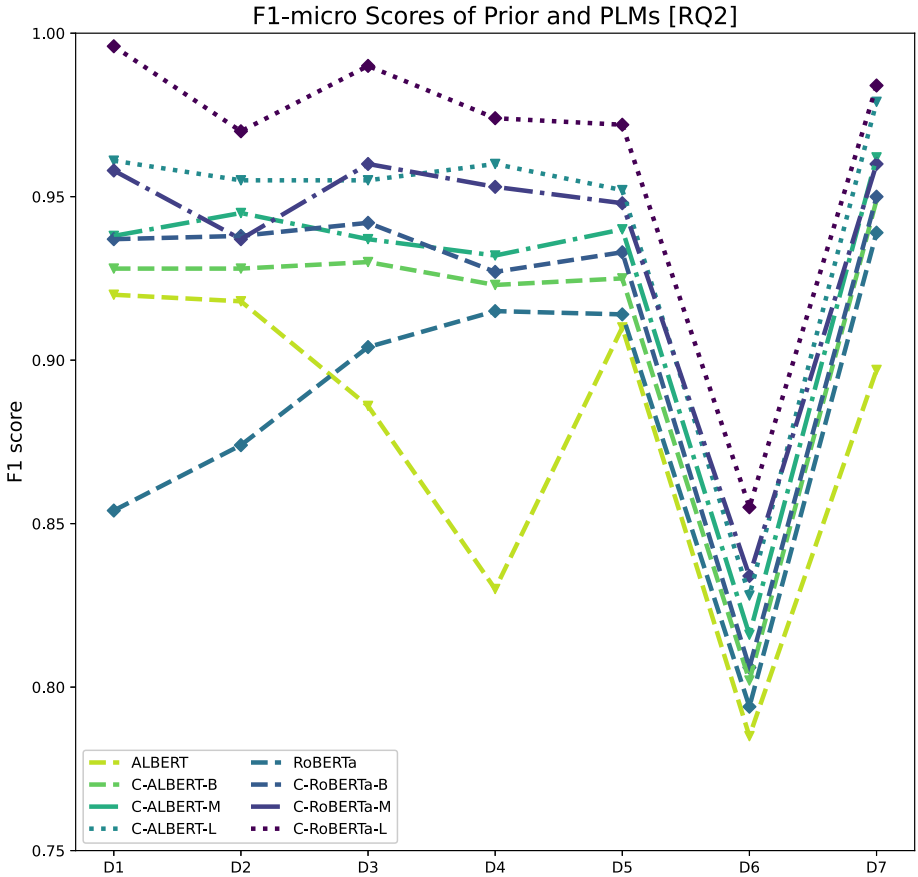


**Fig. 10** F1 micro scores of Custom PLMs (RQ2). The 'BASE,' 'MEDIUM,' and 'LARGE' in the name of the Custom PLMs are abbreviated as 'B,' 'M,' and 'L,' respectively. Similar to the previous results, the trend of all models for the datasets is similar; for example, all models have lower scores for dataset $D_6$. The Custom PLMs improve the scores of the PLMs, and the LARGE Custom models have the best scores among all others

*ALBERT* has lower F1-micro scores for datasets $D_3$ and $D_4$. After pre-training with app reviews, we noticed that all *C-ALBERT* models (i.e., C-ALBERT-BASE, C-ALBERT-MEDIUM, and C-ALBERT-LARGE) have better predictions for both of these datasets ($\sim$ 5% to $\sim$ 13% improved performance). Although this is not the case for $D_6$, all *C-ALBERT* models still improve their performances. We relate this to the low number of reviews per class in $D_6$. Note that the differences between ALBERT and C-ALBERT-BASE on $D_1$ and $D_2$ are not so high; this little boost is related to the pre-training on the smallest portion of the domain-related dataset. However, as the domain-related pre-training dataset's size increased, C-ALBERT models yielded better results. Though for some datasets, the Custom PLM BASE model has a small change, for all cases, the Custom PLMs produce better predictions than their readily available counterpart. *C-RoBERTa* models also have better performance, yielding up to 15.2% increase in performance over *RoBERTa*. As seen in Fig. 10, *C-RoBERTa* improves RoBERTa's F1-micro scores for datasets $D_1$–$D_4$, where *RoBERTa* has scores of below 95.

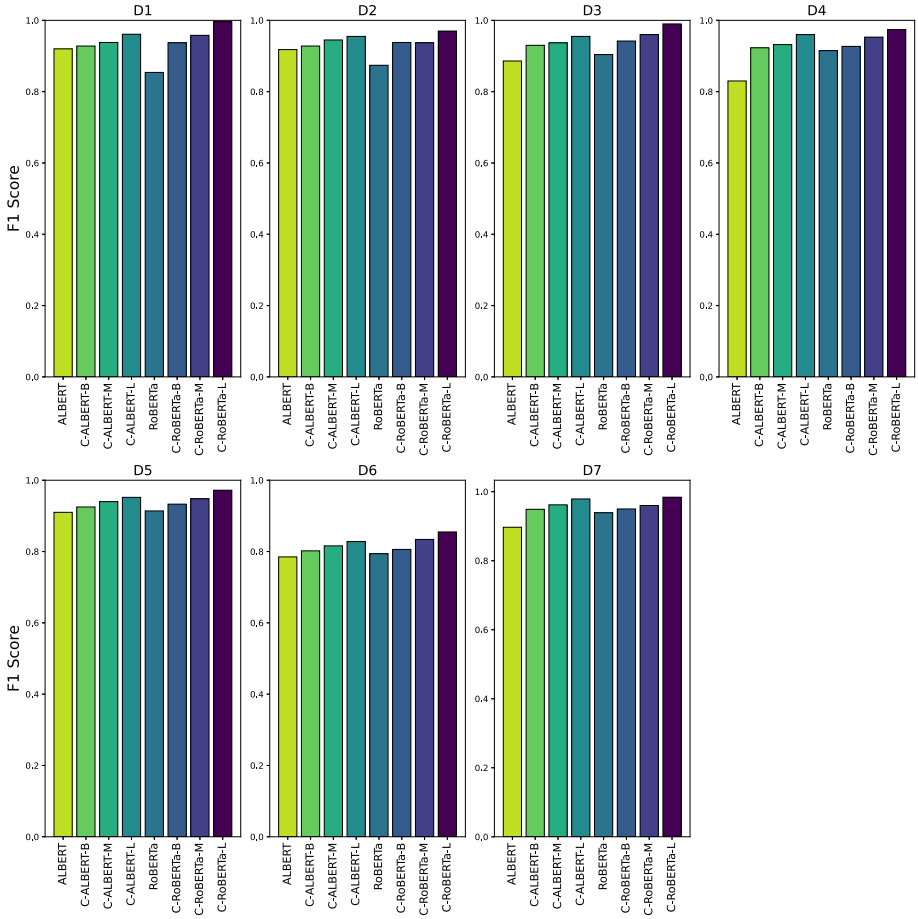


**Fig. 11** F1 micro scores of Custom PLMs (RQ2). The 'BASE,' 'MEDIUM,' and 'LARGE' in the name of the Custom PLMs are abbreviated as 'B,' 'M,' and 'L,' respectively. The scores of all models are shown separately for each dataset. In all datasets, the LARGE Custom PLMs achieve the highest score, and C-RoBERTa-LARGE has the highest F1 score in all datasets

This can be related to the robust pre-training of RoBERTa compared to other PLMs. There is a slight fluctuation in prediction time for all the Custom PLMs.

**Findings of RQ2**

Incorporating domain-specific data (i.e., app reviews) during pre-training of the PLMs improves their performance for app review classification. It slightly improves the prediction time of C-PLMs compared to PLMs. The performance of the models benefits from incorporating more domain-specific data in the pre-training.

## 6.3 RQ3: Experimenting PLMs in Different Settings

### 6.3.1 RQ3-1: Binary vs. Multi-Class Setting

We report the binary classification results in Tables 7 and 8 for RQ3-1. The multi-class classification results of the PLMs and C-PLMs are previously reported in Tables 3, 4, 5 and 6. The micro- and macro-scores were calculated to accommodate the existence of multi-class in our datasets, which is no longer the case after we converted each multi-class dataset into binary class datasets. The randomly selected classes for $D_1$ to $D_7$ are *Bug-Report, Inquiry, Reliability, Feature Requests, App Problem, Bug-Report, and Aspect Evaluation*, respectively. These datasets' other classes are labeled as *Other* or *Irrelevant*.

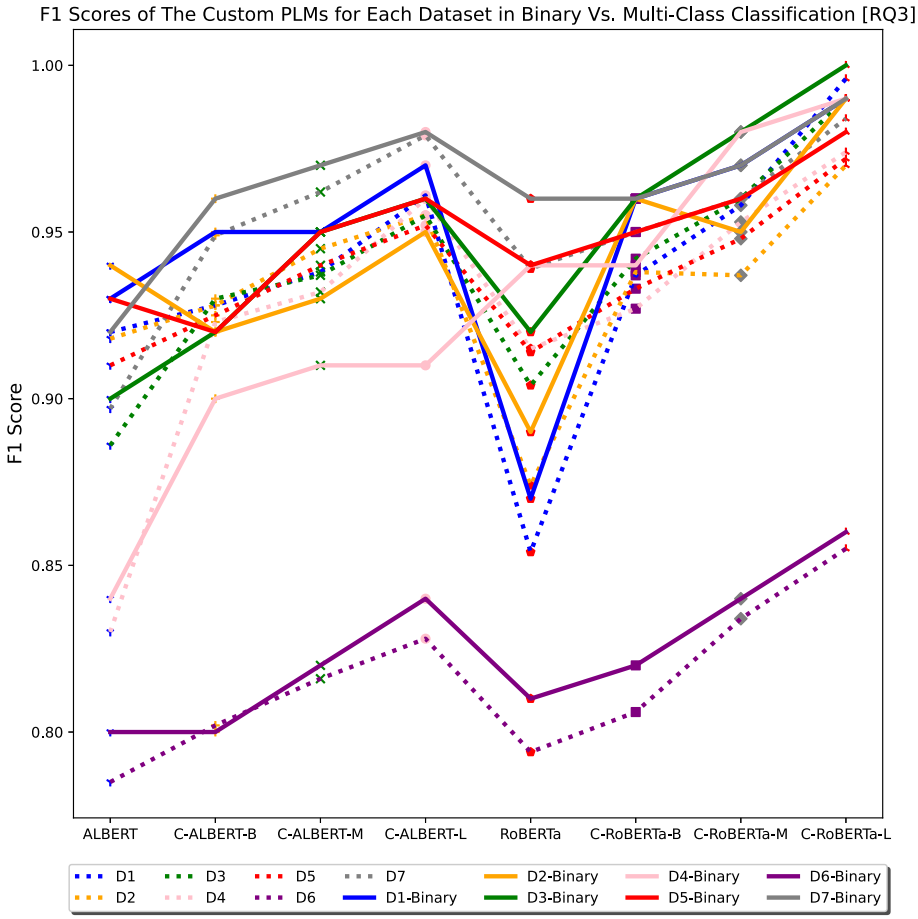


**Fig. 12** F1 micro scores of the ALBERT and RoBERTa based models for binary classification setting (RQ3). The solid lines represent the results of the binary version of each dataset, whereas the dashed lines show the scores of the original datasets, $D_1$–$D_7$, which were reported in RQ1. For most datasets, the models' scores are higher in the binary setting

Figure 12 demonstrates the micro F1 scores of the ALBERT-based and RoBERTa-based models for all datasets. The solid lines in the figure represent the results for the binary version of each dataset, and the dashed lines are the scores for the multi-class classification reported in RQ1. To avoid cluttering, we did not plot the Prior scores in this figure, but all numbers are reported in Tables 7 and 8. From Fig. 12 and the results in Tables 7 and 8 we observe that both *Prior* and *PLM* approaches have performed better in binary classification compared to multi-class classifications. The improvement for the *Prior* approaches ranges from 0 (for *AR-Miner*) to ∼3.2%. On the other hand, the improvement for the *PLM* approaches ranges from 1 to ∼7.7%. All the PLM approaches yielded better performance for binary classification than they did for multi-class classifications. We observe that readily-available PLMs' performance boost is similar to the performance boost produced by the Custom PLMs.

The prediction time of the models for all seven datasets in the binary classification setting is shown in Fig. 18 (see Section Appendix). As the prediction times for XLNET are much higher than the other models, XLNET is not shown in this plot. The Custome ALBERT-based models have the lowest prediction times for datasets $D_4$, $D_5$, and $D_7$, and in par results for $D_6$. Similar to RQ2, the RoBERTa-based models have higher prediction times than ALBERT-based models. These attributes can be related to the dataset's size and the models' architecture.

#### 6.3.2 RQ3-2: Zero-Shot Setting

We report *Precision*, *Recall*, and *F1 scores* for zero-shot classification results in Table 9. As in this setting, we are not fine-tuning our model on a portion of our considered Datasets $D_1$ to $D_7$ (instead, we are fine-tuning our PLMs on a separate NLI dataset), we no longer require cross-validation over multiple folds. The F1 scores are presented in Fig. 13, where each line shows the performance of a model over datasets $D_1$–$D_7$. The 'LARGE' in the model names is abbreviated as 'L' in the plot.

In this setting, we anticipate lower performance because our PLM models do not have any exposure to any instances of the datasets to fine-tune. In this experiment, we use AR-Miner as the baseline to compare the results of this setting for PLMs and C-PLMs. AR-Miner is chosen as it is a topic modeling-based approach that does not require seeing the labelS beforehand. The other models from Prior approaches are not considered here as they are supervised machine learning or deep learning techniques that require training with the labels. The *C-RoBERTa-LARGE-NLI* and *C-ALBERT-LARGE-NLI* are the C-PLMs that are pre-trained on 10 million app reviews (See RQ2). Overall, in $D_7$, *RoBERTa-LARGE-MNLI* yielded the best performance (71%). We observe that RoBERTa-based models achieve higher scores compared to ALBERT-based PLMs. Results of AR-Miner vary between 36% and 44% F1 score for the datasets, and AR-Miner has closer results to *ALBERT-NLI*. For $D_5$, *ALBERT-NLI* yielded a ∼37% F1 score, which is ∼5% worse than the performance of AR-Miner. For Datasets $D_1$ to $D_7$, the PLMs yield on an average 45.875%, 44.425%, 43.125%, 43.225%, 46.45%, 38.8%, 42.725% less F1 score than their counterparts in RQ1. All PLM approaches outperformed AR-Miner, the only Prior approach capable of zero-shot classification, except in $D_5$ for *ALBERT-NLI*. The Custom PLMs, in all cases, improve the results by approximately 10 F1 scores in all datasets. Similar to previous RQs, the ALBERT-based model has a lower prediction time. The prediction times of all models are plotted in Fig. 19 (see Section Appendix).

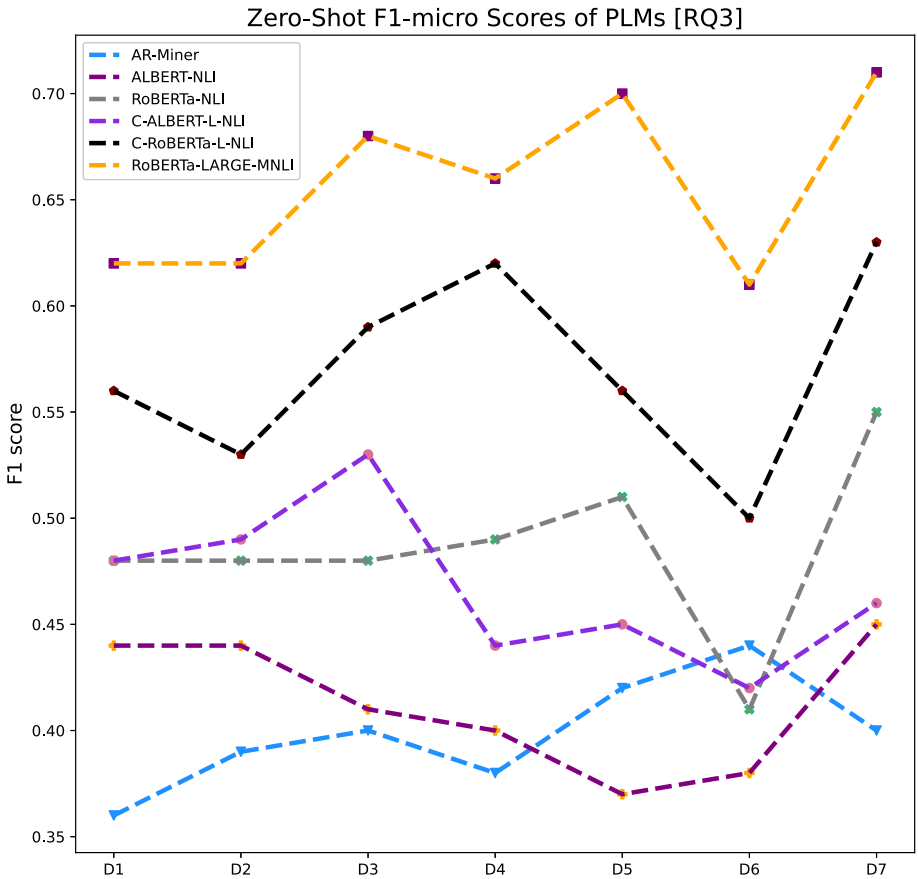


**Fig. 13** The F1 micro scores of AR-Miner, ALBERT, and RoBERTa based models in zero-shot setting (RQ3). The 'LARGE' in the model names is abbreviated and shown as 'L' in C-ALBERT-L-NLI and C-RoBERTa-L-NLI. The best scores belong to the RoBERTa-LARGE-MNLI, which is a ready available PLM trained on NLI data, followed by the LARGE Custom RoBERTa model. Note that the performance of all models is degraded in this setting, compared to when the models are trained on app review classification data

### 6.3.3 RQ3-3: Multi-Task Setting

In Tables 10 and 11, we report the micro and macro *Precision*, *Recall*, and *F1 score* and the training and prediction times for two classification tasks in the app review domain: category classification and sentiment classification. The micro F1 scores of all models are shown in the left part of Fig. 14. The results for sentiment classification are shown in blue and the orange line represents the scores for category classification.

Among all the PLMs, C-ALBERT-LARGE yielded the best result for category classification tasks with micro-average F1 scores of ∼91% and C-RoBERTa-LARGE yielded the best result for sentiment classification task with micro-average F1 scores of ∼93%, respectively.

BERT, ALBERT, RoBERTa, and XLNET performed 2.4%, 1.8%, 7.5%, and 4.3% worse for category classification, respectively; they performed 1.1%, 0.1%, 5.5%, and 0.9% worse for Sentiment classification task, respectively with respect to their averaged micro-F1 scores over seven datasets, in Table 4. C-ALBERT-BASE, MEDIUM, and LARGE performed 4%,

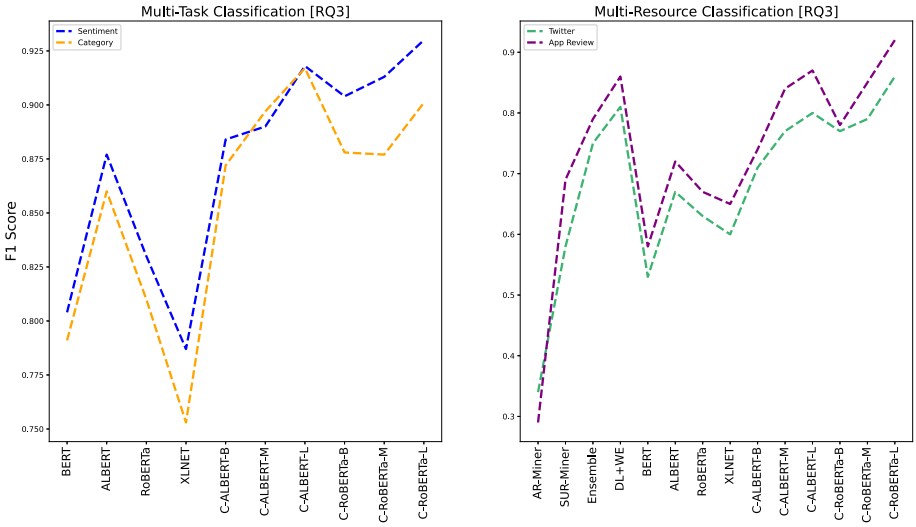


**Fig. 14** F1 micro scores of the models in the multi-task (left) and multi-resource (right) settings (RQ3). *left:* The sentiment classification is shown in blue and the orange line is related to the category classification. The LARGE Custom PLMs, C-ALBERT-LARGE, and C-RoBERTa-LARGE yield the best scores for category and sentiment classifications, respectively. *right:* The F1 scores of the Prior, PLM, and Custom PLMs in the multi-resource setting (RQ3). 'BASE,' 'MEDIUM,' and 'LARGE' are abbreviated as 'B,' 'M,' and 'L' in the model names. The purple line shows the performance of the models when they are trained on Twitter data and tested on App reviews data. The green line is the performance of the models when they are trained on app reviews and tested on Twitter data. For all models, the training on Twitter benefits the performance increase compared to testing on Twitter. The Ensemble and DL+WE models achieve high scores in this setting, though their F1 scores are lower than C-RoBERTa-LARGE model

2.7%, and 2.4% worse for category classification, respectively, compared to their averaged micro-F1 scores over seven datasets, in Table 4; they performed 2.8%, 3.4%, and 2.3% worse for the sentiment classification task, respectively with respect to their F1 scores averaged over seven datasets, in Table 5. On the other hand, C-RoBERTa-BASE, MEDIUM, and LARGE performed 4.1%, 5.9%, and 6.2% worse for category classification, respectively; they performed 1.5%, 2.3%, and 3.3% worse for the sentiment classification task, respectively with respect to their F1 scores averaged over seven datasets, in Table 6. For both classification tasks, we observe that the custom PLMs achieve higher scores than the PLMs, showing that although the task is different (e.g., sentiment analysis vs. app review classification), they still benefit from pre-training the models on the same domain (app reviews). Similar to previous results, Custom PLMs benefit from more app review training data, and the LARGE Custom PLMs achieve higher scores.

### 6.3.4 RQ3-4: Multi-Resource Setting

Dataset $D_2$ contains labeled app reviews from two different sources: App Store and Twitter. In the previous results (Tables 3 to 10), we have not incorporated app reviews collected from Twitter. Here, we use App Store and Twitter reviews from $D_2$.

**Training on App Store App-reviews and Testing on Twitter** In the first part of this setting, we fine-tuned all the available approaches (except for AR-Miner) on app reviews from App Store and tested them on app reviews collected from a different resource, i.e., Twitter. The green line in Fig. 14 (right plot) shows the F1 scores in this setting and all the results are available in Table 12. As a topic modeling approach, *AR-Miner* does not require supervised training.

C-RoBERTa-LARGE achieves the best score among all the considered approaches for this setting in terms of F1 score (0.86). From the previous approaches, DL+WE approach scored higher than the readily available PLMs, and all Custom PLMs, except C-RoBERTa-LARGE. It also beats the C-ALBERT-LARGE by 1% F1 score. The Ensemble method surpassed only readily available PLMs and closely matched the performance of C-ALBERT-MEDIUM. AR-Miner and SUR-Miner have the lowest scores among the Priors, and we do not observe a considerable decline in their performance concerning the single-resource setting in terms of F1 scores. These approaches have scored (∼5%) and (∼2%) less in multi-resource settings than their single-source counterparts.

Compared to the setting where PLMs are evaluated only on app reviews from App Store, we observe ∼13% to ∼31% performance reduction for micro-F1 scores. This performance reduction is less for Custom-PLMs. Among the Custom ALBERT models, the *C-ALBERT-LARGE* model produced the best micro-F1 score of 0.80 (∼25% reduction), and *C-ALBERT-BASE* model produced the least micro-F1 score of 0.71 (∼21% reduction). The C-RoBERTa group has less reduction compared to C-ALBERT models, where the *C-RoBERTa-LARGE* produced the best micro-F1 score of 0.86 (∼15% reduction) and *C-RoBERTa-BASE* model produced the least micro-F1 score of 0.77 (∼15% reduction) among all C-RoBERTa models.

**Training on Twitter and Testing on App Store App-reviews** In the second part of this setting, we trained/fine-tuned all the available approaches (except for AR-Miner) on Twitter and tested them on app reviews collected from a different resource, App Store. The results are available in Table 13 and the F1 scores of this setting are shown in purple line in the right part of Fig. 14.

C-RoBERTa-LARGE achieves the best score among all the considered approaches for this setting, with an F1 score of 0.92. From the previous approaches, the DL+WE approach scored higher than the readily available PLMs in this setting. It could not surpass the C-ALBERT-LARGE but beat the C-ALBERT-MEDIUM by 1% F1 score. The Ensemble method outperformed only readily available PLMs and C-ALBERT-BASE from the Custom PLMs. AR-Miner and SUR-Miner have the lowest scores among the Priors, which have scored (∼10%) less and (∼2%) more in multi-resource settings than their single-source counterparts, respectively. SUR-Miner beat the BERT in this setting by ∼4%.

Compared to the setting where PLMs are evaluated only on app reviews from App Store, we observe ∼9% to ∼30% performance reduction for micro-F1 scores. This performance reduction is less for Custom-PLMs. Among the Custom ALBERT models, the *C-ALBERT-LARGE* model produced the best micro-F1 score of 0.87 (∼8% reduction), and *C-ALBERT-BASE* model produced the least micro-F1 score of 0.74 (∼18% reduction). The C-RoBERTa group has less reduction compared to C-ALBERT models, where the *C-RoBERTa-LARGE* produced the best micro-F1 score of 0.92 (∼5% reduction) and *C-RoBERTa-BASE* model produced the least micro-F1 score of 0.78 (∼14% reduction) among all C-RoBERTa models. The smaller performance reduction for the models in this part, compared to when models are trained on App Store data and tested on Twitter data, could be related to the size of the training data. The dataset $D_2$ has a higher number of labeled tweets (∼ 10K) than the app reviews from the App Store (∼ 6.5K).

Figure 15 demonstrates the prediction time of the models in this setting, as well as for the multi-class and binary classification of dataset $D_2$, (shown in RQ1, RQ2, and RQ3-1). The horizontal lines show the minimum prediction time of the Prior approaches, which belongs to DL+WE model. From the plot, we observe that the prediction times of all Custom PLMs are lower than Prior approaches in this setting.

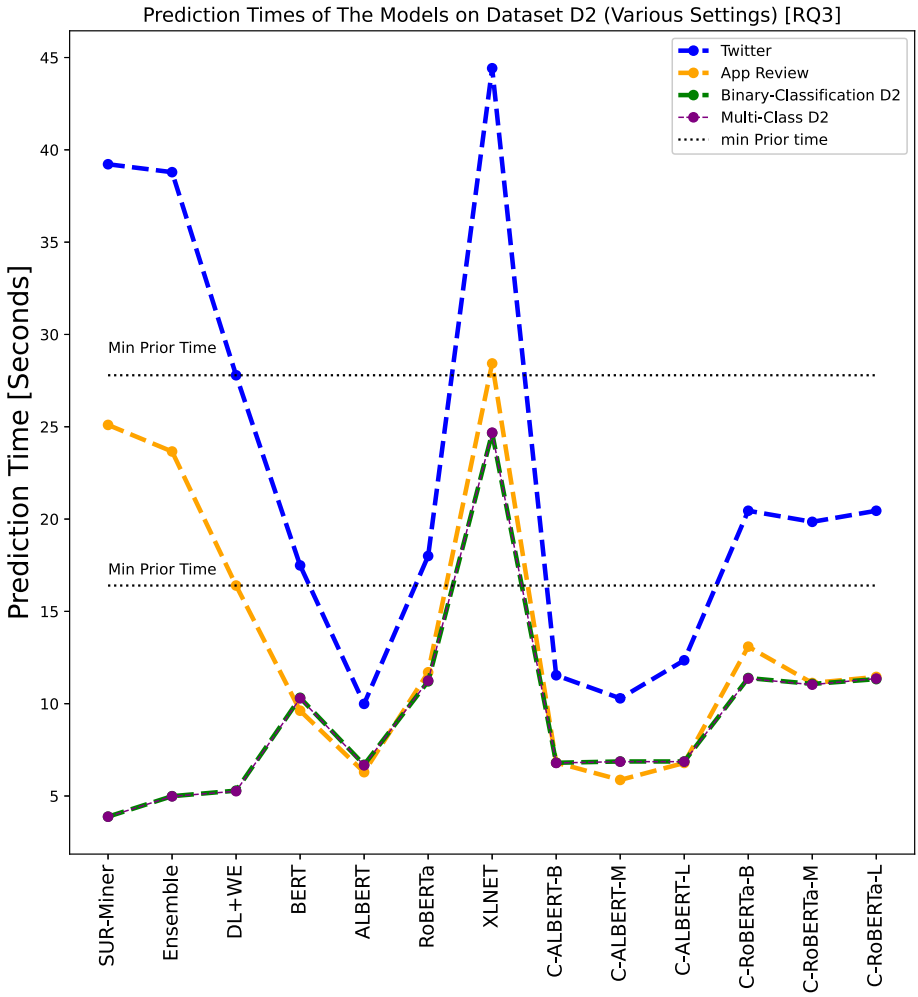


**Fig. 15** The prediction times of all models for dataset $D_2$ (RQ3). The orange line shows the prediction times when models are trained on Twitter data and tested on the app review dataset. The blue line shows the times when the models are trained on app review and tested on Twitter data. The horizontal lines show the minimum prediction times of the Prior approaches. The prediction times of all Custom PLMs are lower than the Prior approaches (horizontal lines). The other two lines in green and purple show the prediction times of models on dataset $D_2$ according to RQ1, RQ2, and RQ3-1, where the models are evaluated in multi-class and binary settings. Note that the prediction times for multi-class and binary classifications are approximately the same

**Findings of RQ3**

1. Both Prior and PLM models yield better performance for binary classification tasks than for multi-class settings.
2. PLMs are the best choice for zero-shot classification setting, and Custom PLMs, though having less score than the large RoBERTa-MNLI model, improve the results of their non-domain specific models by approximately 10 F1 scores.
3. For multi-task and multi-resource settings, the readily available PLMs and Custom PLMs showed the same trend as demonstrated in RQ1 and RQ2, that is *Readily Available PLM* $<$ *Custom-BASE* $<$ *Custom-MEDIUM* $<$ *Custom-LARGE*.
4. RoBERTa-based Custom PLMs have the best scores among all models. They also have lower prediction times in the multi-resource setting than Prior approaches.

## 7 Discussions

### 7.1 Implications for Users

In the following, we discuss our findings and implications for the users[12].

**Prior approaches:** ***Deep learning with word embedding can be used for app issue classification when one does not want to use PLMs*** Applying PLMs requires prior knowledge, including fine-tuning and preventing overfitting. Additionally, the computational power required to use them might only be available to some. These prior learning and computational needs might discourage some users from applying PLMs. So, among the Prior approaches, we suggest using the Deep Learning Technique leveraging Word Embedding (DL+WE) as it generated the best results for predicting the correct classes of app reviews (RQ1), followed closely by the Ensemble Method. DL+WE also took the least time to predict the classes, whereas SUR-Miner took the second least time. However, if no labels are available to train the mentioned models, AR-Miner should be used, which utilizes topic modeling to separate the classes of app reviews.

**Prior approaches vs. PLMs:** ***If the prediction time does not matter much, but high performance is required, PLMs should be used to classify app issues*** App issue classification studied in our work is mainly used by mobile app developers to help them analyze their users' feedback. As a result, a small increase in the prediction time might be compromised to achieve a higher performance depending on their needs. From the results of RQ1, we observed that all PLMs outperformed the Prior approaches (except for 1 out of 28 cases) to predict classes accurately. RoBERTa and ALBERT scored the highest F1 scores in most of the datasets among D1 to D7. Compared with the fastest Prior approach, ALBERT has the lowest, and XLNET has the highest increase in prediction time. Consequently, it is worth applying PLMs for app-review classification if the developer, researcher, or software engineer can spare marginally additional time to achieve higher prediction scores.

[12] Here, users can be developers, practitioners, or researchers who want to use the studied models to classify issues related to mobile apps from user feedback.

**Prior approaches vs. PLMs:** ***PLMs can be used for all settings when higher performance is required, except for the multi-resource setting, where the deep learning model with word embedding has better scores*** For the multi-class classification (RQ1), PLMs perform significantly better than the Prior approaches for most datasets. For Dataset $D_2$, the difference in the scores between the two groups is not much, but when we consider $D_7$, the results of PLMs are almost 10 F1 score higher than the best Prior approach. The prediction times also vary, but some PLMs can predict the labels at approximately the same time as the best Prior approaches (Ensemble and DL+WE). For binary classification, the best Prior approaches achieve scores below 90% and mainly in the low 80% F1 score. Nevertheless, the F1 scores of PLMs are mainly above 90%, and the Custom PLMs have even higher scores. However, for the multi-resource setting (RQ3-4), the Ensemble and DL+WE have better F1 scores than all PLMs, but not higher than C-PLMs. This result is interesting, showing that the learned knowledge about the app review domain for issue classification is essential in this setting, whether the model is trained on App Store data and tested on Twitter data or vice versa. This is confirmed by the obtained results of the C-RoBERTa-LARGE, which have the highest scores among all models in this setting.

**Binary vs. multi-class:** ***When possible, the classification task should be turned into a binary classification to achieve higher scores*** All PLMs and Prior approaches performed better in binary classification than in multi-class classification. From Tables 7 and 8, we observe that converting or trimming a multi-class dataset to a binary class dataset helps all the Prior and PLM approaches yield better performance. Prior approaches are improved by 0 to ∼3%, whereas the gain for the PLMs ranges from 1 to ∼7%. All PLM approaches performed better for binary classification than for multi-class classifications. The performance boost produced by readily-available PLMs is proportionate to the performance boost produced by Custom PLMs.

**Zero-shot:** ***In a zero-shot setting, the Roberta-based Custom PLMs or RoBERTa-LARGE-MNLI can be used to classify app issues*** All PLMs can be implemented in a Zero-Shot setting and yield adequate performance. The only Prior approach that can be used directly in this setting is AR-Miner which is based on topic modeling. All the readily available and Custom PLMs are fine-tuned on NLI dataset for Inference classification task. Due to the fact that these PLM models had not been exposed to any instances of the datasets in our study in the zero-shot setting, they are expected to perform worse than their reported performance in Tables 4, 5, and 6. The Custom-PLMs pre-trained with domain-specific knowledge could not outperform the readily available *RoBERTa-LARGE-MNLI*. This can be related to the dataset used for its training, which is a much larger dataset than the NLI datasets we used to train our PLMs. Although the C-PLMs cannot beat the *RoBERTa-LARGE-MNLI* model, it is notable that these C-PLMs are smaller models, and the *RoBERTa-LARGE-MNLI* model is an extensive language model. However, the C-PLMs we studied still achieve significant improvements compared to their general domain PLMs in this setting.

**Custom PLMs vs. PLMs:** ***Custom PLMs are the best models that should be used to achieve better prediction scores with lower prediction times*** Customized PLMs that are pre-trained with the app review data have the best performance in all settings: binary classification, multi-class classification, multi-task, and multi-resource setting. Even for zero-shot settings, they perform better than all the other models (except for RoBERTa-LARGE-MNLI, which is trained on much larger NLI data). The benefit of using Custom PLMs, specifically RoBERTa-based ones, compared to other PLMs, is more evident for the multi-task and multi-resource setting in RQ3-3 and RQ3-4. Interestingly, the PLMs do not perform well in the multi-task and multi-resource settings, while Custom PLMs have the highest scores. The PLMs in these two settings perform lower than the Ensemble and DL+WE models. However, the best Custom

PLMs achieve scores above 90% for multi-task and 86% and 92% F1 scores for multi-resource. Other than achieving the best score, Custom PLMs have the lowest prediction time among all models for multi-class classification, multi-task, and multi-resource settings. They also achieve the lowest prediction time for some binary classification datasets. Our results for RQ3-4 confirm that using over-the-shelf PLMs is not the best option when classifying issues for app reviews in all settings. PLMs can still have good performance in binary and multi-class classification. However, for more challenging scenarios (i.e., zero-shot, multi-task, and multi-resource), pre-training them with domain-specific data increases their performance and reduces their prediction time. Moreover, the more app review data is used in the pre-training of Custom PLMs, the better their performances are.

### 7.2 Implications for Researchers

**Training Custom PLMs:** ***Research for incorporating app review data in the pre-training of the current PLMs is needed*** The inclusion of app reviews in the pre-training corpus allows PLMs to classify user feedback with higher F1 scores. We have pre-trained three models of ALBERT and RoBERTa with three different sizes of corpora. This experiment found that more domain-specific information during the pre-training helps the newly trained PLMs yield better predictions of app reviews. C-ALBERT and C-RoBERTa models can improve the ALBERT and RoBERTa scores for up to ∼13% and ∼14%, respectively. Here, we followed the work of Wada et al. (2020) to incorporate app review data for pre-training of the PLMs. However, there is considerable research on training the PLMs with domain-specific data (Chalkidis et al. 2020; Hakala and Pyysalo 2019). Other approaches can be investigated for domain-specific models that achieve higher scores in the settings of RQ3.

**Multi-class classification and zero-shot settings:** ***Research is needed to build models that perform better in multi-class and zero-shot setting*** Our results showed that the studied models perform better in a balanced binary classification than in an imbalanced multi-class setting. This suggests that more research is needed to develop models that perform better when multiple labels are available, and the dataset is imbalanced. Moreover, researchers can work on techniques to increase the performance of the models when classifying with new labels that the model has not seen previously.

## 8 Threats to Validity

**Internal Validity** A possible internal validity can be related to the obtained results. To mitigate this threat, we use stratified k-fold cross-validation to avoid the bias that might be introduced to the results by the test set. In addition, to mitigate threats to the validity of the results, hyper-parameter values are kept the same for all PLMs and Deep learning models in all the fine-tuning steps. Also, we prevent overfitting by using early stopping and a higher dropout rate. We also keep the alpha and beta values for LDA as reported in AR-Miner. We run all the experiments on a single machine and report the machine configuration to enforce the reproducibility of the results. Furthermore, we consider the same metrics to compare the PLMs with Prior approaches. Along with the generated raw duration, we also provide the time change in percentage compared to the baseline. Our adopted stricter micro and macro-metrics diminish the ambiguity while comparing the considered approaches' associated results.

**Construct Validity** The selection of the Prior approaches and PLMs can pose a validity threat to our study. We identified the four most common approaches as Priors by examining the highly practiced methods, tools, and techniques employed by researchers and

application developers. The considered approaches include different Machine Learning algorithms, Ensemble methods, and Deep Learning Approaches for app issue classification tasks and are selected after conducting a literature review. The PLMs were adopted by following a previous study (Zhang et al. 2020) that conducted an empirical study on PLMs' performance for sentiment analysis in Software Engineering. Additionally, we pre-trained transformer-based models by incorporating app-review-related datasets with the previously used generic dataset to better understand the potential of domain-specific PLMs (`C-PLM-X`) for app issue classification. Another threat to the study can be related to dataset $D_7$. This dataset merges other datasets and has labels borrowed from them. Although we consulted the definitions of the labels from their publishers, there might be a chance that the samples in one group of a dataset are closely related to the samples from another label of another dataset, thus, affecting the results. We mitigate this threat by combining labels from different datasets with similar names. We also used stratified k-fold cross-validation to alleviate potential threats.

**External Validity** In this study, we empirically study the ability of the PLMs in issue classification of app reviews. The app review 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 other tasks, as mentioned in the paper. Our study is therefore limited to app review classification for software engineers, and we do not study PLMs for other purposes such as finding issues from business perspectives (Tang 2019), product reviews (Zhao et al. 2017), and applications such as intention mining (Huang et al. 2020). We also do not study the summarization of relevant issues from app reviews. However, we conduct our study on six different datasets. In addition, by merging them, we run experiments on $D_7$, which has multiple classes covering different aspects of apps useful for app developers. Another threat to the generalizability of the results lies in the different tasks we considered. We mitigate this threat by studying app reviews' sentiment classification and category classification in the multi-task setting. Although PLMs *might* be useful for other tasks, we do not study them in this work. To ensure the generalizability of our study in the specified scope, we have incorporated datasets that include app reviews from diverse app categories and two platforms, i.e., Google Play and Apple App Store. The datasets have various sizes and different labels. In addition, we experiment with issue classification from another platform, i.e., Twitter, which has been shown to require different models than App Stores due to the differences in the platforms, noise data, and user feedback.

## 9 Conclusion

We conducted an extensive exploratory study comparing app issue classification tools and pre-trained Transformer-based models in various settings. We conducted the experiments on six available datasets and a highly imbalanced dataset, a combination of the six datasets. Domain-specific PLMs were trained using different sizes of app review data we collected from Google Play, and these customized PLMs were also studied here. Our results confirm that PLMs achieve higher scores in binary and multi-class classification than Prior approaches. However, the over-the-shelf PLMs are not always the best models to be used in all scenarios. Instead, C-PLMs have the highest scores and can perform better than other models in all settings. Moreover, incorporating app-specific data in the pre-training of PLMs reduces the prediction time. One of the future directions of this research is assessing domain-specific PLMs in other areas of app reviews and exploring ways to increase performance in zero-shot settings.

**Acknowledgements** This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada RGPIN-2019-05175.

## Appendix

**Table 3** Performance and efficiency of the Prior approaches (RQ1)

| | Micro-avg Scores | | | Macro-avg scores | | | Training Time (s) | Prediction Time (s) |
|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | | |
| Dataset-1 | | | | | | | | |
| AR-MINER | 0.36 | 0.36 | 0.36 | No micro or macro average | | | 261.71 | NA |
| SUR-MINER | 0.57 | 0.56 | 0.56 | 0.53 | 0.47 | 0.5 | 105.31 | 21.06 |
| Ensemble | 0.71 | 0.7 | 0.7 | 0.7 | 0.6 | 0.65 | 115.84 | 23.17 |
| DL+WE | **0.84** | **0.71** | **0.77** | **0.83** | **0.67** | **0.74** | 99.62 | **14.23** |
| Dataset-2 | | | | | | | | |
| AR-MINER | 0.49 | 0.32 | 0.39 | No micro or macro average | | | 81.17 | NA |
| SUR-MINER | 0.6 | 0.55 | 0.57 | 0.65 | 0.51 | 0.57 | 27.18 | **3.88** |
| Ensemble | 0.8 | 0.69 | 0.74 | 0.7 | 0.66 | 0.68 | 29.9 | 4.98 |
| DL+WE | **0.93** | **0.84** | **0.88** | **0.94** | **0.84** | **0.89** | 26.31 | 5.26 |
| Dataset-3 | | | | | | | | |
| AR-MINER | 0.44 | 0.36 | 0.4 | No micro or macro average | | | 74.62 | NA |
| SUR-MINER | 0.62 | 0.63 | 0.62 | 0.59 | 0.53 | 0.56 | 19.33 | **3.22** |
| Ensemble | 0.77 | 0.75 | 0.76 | 0.69 | 0.71 | 0.7 | 21.46 | 4.29 |
| DL+WE | **0.86** | **0.84** | **0.85** | **0.77** | **0.72** | **0.74** | 19.74 | 3.95 |
| Dataset-4 | | | | | | | | |
| AR-MINER | 0.45 | 0.33 | 0.38 | No micro or macro average | | | 71.96 | NA |
| SUR-MINER | 0.63 | 0.58 | 0.6 | 0.55 | 0.53 | 0.54 | 11.74 | 2.35 |
| Ensemble | 0.84 | 0.74 | 0.79 | 0.82 | 0.7 | 0.76 | 13.5 | 2.7 |
| DL+WE | **0.87** | **0.76** | **0.81** | **0.85** | **0.73** | **0.79** | 12.29 | **2.05** |
| Dataset-5 | | | | | | | | |
| AR-MINER | 0.53 | 0.35 | 0.42 | No micro or macro average | | | 57.02 | NA |
| SUR-MINER | 0.62 | 0.54 | 0.58 | 0.57 | 0.56 | 0.56 | 9.11 | **1.3** |
| Ensemble | **0.86** | **0.76** | **0.81** | **0.84** | **0.81** | **0.82** | 10.93 | 2.19 |
| DL+WE | 0.84 | 0.75 | 0.79 | 0.83 | **0.81** | **0.82** | 9.73 | 1.62 |
| Dataset-6 | | | | | | | | |
| AR-MINER | 0.47 | 0.41 | 0.44 | No micro or macro average | | | 71.91 | NA |
| SUR-MINER | 0.58 | 0.52 | 0.55 | 0.6 | 0.57 | 0.58 | 10.07 | **1.44** |
| Ensemble | 0.75 | 0.73 | 0.74 | 0.85 | 0.81 | 0.83 | 11.98 | 1.71 |
| DL+WE | **0.77** | **0.76** | **0.76** | **0.86** | **0.82** | **0.84** | 10.3 | 1.47 |
| Dataset-7 | Merged Dataset | | | | | | | |
| AR-MINER | 0.49 | 0.34 | 0.4 | No micro or macro average | | | 401.57 | NA |
| SUR-MINER | 0.66 | 0.55 | 0.6 | 0.64 | 0.6 | 0.62 | 156.54 | 31.31 |
| Ensemble | 0.76 | 0.7 | 0.73 | 0.79 | **0.76** | 0.77 | 176.89 | 29.48 |
| DL+WE | **0.84** | **0.78** | **0.81** | **0.82** | 0.74 | **0.78** | 155.66 | **25.94** |

For each model, the Micro- and Macro-average scores for Precision (P), Recall (R) and F1 scores (F1) and the training and prediction times in seconds (s) are reported. The best scores for each dataset are shown in bold. DL+WE generated the best scores on most datasets, with the lowest prediction time for $D_7$

**Table 4** Performance and efficiency of the pre-trained models (RQ1)

| | Micro-avg Scores | | | Macro-avg scores | | | Training Time (s) | Prediction Time (s) | Time-diff (%) |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | | | |
| Dataset-1 | | | | | | | | | |
| BERT | 0.79 | 0.74 | 0.764 | 0.76 | 0.69 | 0.723 | 345.8 | 19.21 | 34.34 |
| ALBERT | **0.93** | **0.91** | **0.92** | **0.89** | **0.84** | **0.864** | 286.5 | **15.92** | 11.33 |
| RoBERTa | 0.89 | 0.82 | 0.854 | 0.85 | 0.82 | 0.835 | 357.35 | 19.85 | 38.81 |
| XLNET | 0.76 | 0.68 | 0.718 | 0.72 | 0.68 | 0.699 | 514.92 | 36.78 | 157.2 |
| Dataset-2 | | | | | | | | | |
| BERT | 0.77 | 0.69 | 0.728 | 0.86 | 0.83 | 0.845 | 185.14 | 10.29 | 165.21 |
| ALBERT | **0.96** | **0.88** | **0.918** | **0.94** | **0.89** | **0.914** | 119.93 | **6.66** | 71.65 |
| RoBERTa | 0.9 | 0.85 | 0.874 | 0.79 | 0.74 | 0.764 | 202.44 | 11.25 | 189.95 |
| XLNET | 0.85 | 0.77 | 0.808 | 0.87 | 0.78 | 0.823 | 345.57 | 24.68 | 536.08 |
| Dataset-3 | | | | | | | | | |
| BERT | **0.93** | **0.91** | **0.92** | **0.9** | **0.82** | **0.858** | 156.9 | 6.82 | 111.8 |
| ALBERT | 0.91 | 0.87 | 0.89 | 0.89 | 0.79 | 0.837 | 105.53 | **4.59** | 42.55 |
| RoBERTa | **0.93** | 0.88 | 0.904 | 0.89 | 0.81 | 0.848 | 175.29 | 7.62 | 136.65 |
| XLNET | 0.88 | 0.86 | 0.87 | 0.79 | 0.73 | 0.759 | 307.66 | 17.09 | 430.75 |
| Dataset-4 | | | | | | | | | |
| BERT | 0.84 | 0.78 | 0.809 | 0.79 | 0.72 | 0.753 | 75.69 | 2.91 | 41.95 |
| ALBERT | 0.85 | 0.81 | 0.83 | 0.72 | 0.65 | 0.683 | 53.66 | **2.06** | 0.49 |
| RoBERTa | **0.93** | **0.9** | **0.915** | **0.87** | **0.81** | **0.839** | 87.61 | 3.37 | 64.39 |
| XLNET | 0.78 | 0.75 | 0.765 | 0.8 | 0.76 | 0.779 | 149.36 | 7.11 | 246.83 |
| Dataset-5 | | | | | | | | | |
| BERT | 0.88 | 0.81 | 0.844 | 0.8 | 0.77 | 0.785 | 74.52 | 2.57 | 97.69 |
| ALBERT | 0.93 | **0.89** | 0.91 | **0.93** | **0.88** | **0.904** | 38.63 | **1.33** | 2.31 |

**Table 4** continued

| | Micro-avg Scores | | | Macro-avg scores | | | Training Time (s) | Prediction Time (s) | Time-diff (%) |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | | | |
| RoBERTa | **0.94** | **0.89** | **0.914** | 0.87 | 0.83 | 0.85 | 80.69 | 2.78 | 113.85 |
| XLNET | 0.87 | 0.79 | 0.828 | 0.924 | 0.85 | **0.904** | 127.46 | 4.9 | 276.92 |
| Dataset-6 | | | | | | | | | |
| BERT | 0.79 | 0.74 | 0.764 | 0.84 | 0.82 | 0.83 | 74.83 | 2.77 | 92.36 |
| ALBERT | 0.8 | **0.77** | 0.785 | **0.93** | 0.86 | 0.894 | 48.74 | **1.81** | 25.69 |
| RoBERTa | **0.82** | **0.77** | **0.794** | **0.93** | **0.91** | **0.92** | 81.2 | 3.01 | 109.03 |
| XLNET | 0.73 | 0.66 | 0.693 | **0.93** | 0.85 | 0.888 | 143.15 | 6.22 | 331.94 |
| Dataset-7 | Merged Dataset | | | | | | | | |
| BERT | 0.91 | 0.84 | 0.874 | 0.88 | 0.8 | 0.844 | 574.32 | 24.97 | -3.74 |
| ALBERT | 0.95 | 0.85 | 0.897 | 0.85 | 0.8 | 0.824 | 460.19 | **20.01** | -22.86 |
| RoBERTa | **0.97** | **0.91** | **0.939** | 0.87 | 0.82 | 0.838 | 597.65 | 25.98 | 0.15 |
| XLNET | 0.92 | 0.86 | 0.889 | **0.89** | **0.84** | **0.864** | 836.96 | 49.23 | 89.78 |

For each model, the Micro- and Macro-average scores for Precision (P), Recall (R), and F1 scores (F1) and the training and prediction times in seconds (s) are reported. The last column on the right represents the time differences in the percentage of PLMs compared to the Prior approaches. The best scores for each dataset are shown in bold. ALBERT and RoBERTa achieve the best scores for most datasets, ALBERT having the lowest prediction time

**Table 5** RQ2 results: Performance of Custom pre-trained models (i.e., trained on app reviews data), shown by 'C' in front of the model name

| | Micro-avg Scores | | | Macro-avg scores | | | Training Time (s) | Prediction Time (s) | Time-Diff (%) |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | | | |
| Dataset-1 | | | | | | | | | |
| ALBERT (PLM-X1) | 0.93 | 0.91 | 0.92 | 0.89 | 0.84 | 0.864 | 286.5 | 15.92 | 11.33 |
| C-ALBERT-BASE | 0.94 | 0.916 | 0.928 | 0.902 | 0.844 | 0.872 | 295.1 | 15.6 | 9.09 |
| C-ALBERT-MEDIUM | 0.95 | 0.926 | 0.938 | 0.908 | 0.865 | 0.886 | 280.77 | **15.44** | 7.97 |
| C-ALBERT-LARGE | **0.977** | **0.945** | **0.961** | **0.92** | **0.87** | **0.894** | 295.1 | 15.6 | 9.09 |
| Dataset-2 | | | | | | | | | |
| ALBERT (PLM-X1) | 0.96 | 0.88 | 0.918 | 0.94 | 0.89 | 0.914 | 119.93 | **6.66** | 71.65 |
| C-ALBERT-BASE | 0.972 | 0.888 | 0.928 | 0.947 | 0.894 | 0.92 | 117.53 | 6.79 | 75 |
| C-ALBERT-MEDIUM | 0.987 | 0.906 | 0.945 | 0.958 | 0.903 | 0.93 | 123.53 | 6.86 | 76.8 |
| C-ALBERT-LARGE | **0.991** | **0.91** | **0.955** | **0.978** | **0.933** | **0.955** | 121.13 | 6.86 | 76.8 |
| Dataset-3 | | | | | | | | | |
| ALBERT (PLM-X1) | 0.91 | 0.87 | 0.886 | 0.8 | 0.75 | 0.774 | 105.53 | 4.59 | 42.55 |
| C-ALBERT-BASE | 0.938 | 0.922 | 0.93 | 0.91 | 0.829 | 0.868 | 107.64 | 4.64 | 44.1 |
| C-ALBERT-MEDIUM | 0.949 | 0.926 | 0.937 | 0.914 | 0.841 | 0.876 | 106.59 | **4.5** | 39.75 |
| C-ALBERT-LARGE | **0.965** | **0.945** | **0.955** | **0.941** | **0.845** | **0.89** | 104.47 | 4.64 | 44.1 |
| Dataset-4 | | | | | | | | | |
| ALBERT (PLM-X1) | 0.85 | 0.81 | 0.83 | 0.72 | 0.65 | 0.683 | 53.66 | 2.06 | 0.49 |
| C-ALBERT-BASE | 0.94 | 0.907 | 0.923 | 0.875 | 0.821 | 0.847 | 54.2 | **2.02** | -1.46 |
| C-ALBERT-MEDIUM | 0.945 | 0.92 | 0.932 | 0.886 | 0.832 | 0.858 | 53.12 | 2.04 | -0.49 |
| C-ALBERT-LARGE | **0.976** | **0.945** | **0.96** | **0.9** | **0.841** | **0.87** | 52.59 | 2.08 | 1.46 |
| Dataset-5 | | | | | | | | | |
| ALBERT (PLM-X1) | 0.93 | 0.89 | 0.91 | 0.93 | 0.88 | 0.904 | 38.63 | 1.33 | 2.31 |
| C-ALBERT-BASE | 0.952 | 0.899 | 0.925 | 0.942 | 0.892 | 0.916 | 39.4 | **1.3** | 0 |

**Table 5** continued

| | Micro-avg Scores | | | Macro-avg scores | | | Training Time (s) | Prediction Time (s) | Time-Diff (%) |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | | | |
| C-ALBERT-MEDIUM | 0.965 | 0.917 | 0.94 | 0.957 | 0.898 | 0.927 | 37.86 | **1.3** | 0 |
| C-ALBERT-LARGE | **0.984** | **0.922** | **0.952** | **0.967** | **0.908** | **0.937** | 38.24 | 1.34 | 3.08 |
| Dataset-6 | | | | | | | | | |
| ALBERT (PLM-X1) | 0.8 | 0.77 | 0.785 | 0.93 | 0.86 | 0.894 | 48.74 | 1.81 | 25.69 |
| C-ALBERT-BASE | 0.831 | 0.775 | 0.802 | 0.938 | 0.921 | 0.929 | 49.71 | 1.83 | 27.08 |
| C-ALBERT-MEDIUM | 0.842 | 0.791 | 0.816 | 0.95 | 0.924 | 0.937 | 49.71 | **1.77** | 22.92 |
| C-ALBERT-LARGE | **0.86** | **0.799** | **0.828** | **0.963** | **0.943** | **0.953** | 49.71 | 1.83 | 27.08 |
| Dataset-7 | Merged Dataset | | | | | | | | |
| ALBERT (PLM-X1) | 0.95 | 0.85 | 0.897 | 0.85 | 0.8 | 0.824 | 460.19 | 20.01 | -22.86 |
| C-ALBERT-BASE | 0.98 | 0.92 | 0.949 | 0.903 | 0.848 | 0.875 | 464.79 | 20.21 | -22.09 |
| C-ALBERT-MEDIUM | 0.993 | 0.933 | 0.962 | 0.908 | 0.86 | 0.883 | 450.99 | 20.41 | -21.32 |
| C-ALBERT-LARGE | **0.998** | **0.942** | **0.979** | **0.935** | **0.866** | **0.899** | 469.39 | **19.81** | -23.63 |

The BASE, MEDIUM, and LARGE models are related to the number of app reviews used in the pre-training. The Micro- and Macro-average scores for Precision (P), Recall (R) and F1 scores (F1) and the training and prediction times in seconds (s) are reported. The last column on the right represents the time differences in percentage for PLMs compared to the Prior approaches. The best scores for each dataset are shown in bold. C-ALBERT-LARGE achieves the highest score, and lower prediction time for $D_7$

**Table 6** RQ2 results: Performance of Custom pre-trained models (i.e., trained on app reviews data), shown by 'C' in front of the model name

| | Micro-avg Scores | | | Macro-avg scores | | | Training Time (s) | Prediction Time (s) | Time Diff (%) |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | | | |
| Dataset-1 | | | | | | | | | |
| RoBERTa (PLM-X2) | 0.89 | 0.82 | 0.854 | 0.85 | 0.82 | 0.835 | 357.35 | 19.85 | 38.81 |
| C-RoBERTa-BASE | 0.949 | 0.925 | 0.937 | 0.903 | 0.85 | 0.876 | 360.92 | 19.65 | 37.41 |
| C-RoBERTa-MEDIUM | 0.958 | 0.959 | 0.958 | 0.933 | 0.869 | 0.9 | 360.92 | 20.25 | 41.61 |
| C-RoBERTa-LARGE | **0.991** | **0.982** | **0.996** | **0.97** | **0.911** | **0.94** | 364.5 | **19.45** | 36.01 |
| Dataset-2 | | | | | | | | | |
| RoBERTa (PLM-X2) | 0.9 | 0.85 | 0.874 | 0.79 | 0.74 | 0.764 | 202.44 | 11.25 | 189.95 |
| C-RoBERTa-BASE | 0.985 | 0.895 | 0.938 | 0.949 | 0.902 | 0.925 | 204.46 | 11.36 | 192.78 |
| C-RoBERTa-MEDIUM | 0.962 | 0.914 | 0.937 | 0.989 | 0.938 | 0.963 | 200.42 | **11.03** | 184.28 |
| C-RoBERTa-LARGE | **0.985** | **0.956** | **0.97** | **0.997** | **0.946** | **0.971** | 204.46 | 11.36 | 192.78 |
| Dataset-3 | | | | | | | | | |
| RoBERTa (PLM-X2) | 0.93 | 0.88 | 0.904 | 0.89 | 0.81 | 0.848 | 175.29 | 7.62 | 136.65 |
| C-RoBERTa-BASE | 0.953 | 0.931 | 0.942 | 0.925 | 0.845 | 0.883 | 173.54 | 7.7 | 139.13 |
| C-RoBERTa-MEDIUM | 0.978 | 0.942 | 0.96 | 0.927 | 0.859 | 0.892 | 177.04 | **7.47** | 131.99 |
| C-RoBERTa-LARGE | **0.995** | **0.986** | **0.99** | **0.965** | **0.884** | **0.923** | 177.04 | **7.47** | 131.99 |
| Dataset-4 | | | | | | | | | |
| RoBERTa (PLM-X2) | 0.93 | 0.9 | 0.915 | 0.87 | 0.81 | 0.839 | 87.61 | **3.37** | 64.39 |
| C-RoBERTa-BASE | 0.943 | 0.912 | 0.927 | 0.879 | 0.82 | 0.848 | 88.49 | 3.44 | 67.8 |
| C-RoBERTa-MEDIUM | 0.966 | 0.94 | 0.953 | 0.897 | 0.841 | 0.868 | 89.36 | 3.4 | 65.85 |
| C-RoBERTa-LARGE | **0.986** | **0.962** | **0.974** | **0.947** | **0.877** | **0.911** | 89.36 | 3.4 | 65.85 |
| Dataset-5 | | | | | | | | | |
| RoBERTa (PLM-X2) | 0.94 | 0.89 | 0.914 | 0.87 | 0.83 | 0.85 | 80.69 | 2.78 | 113.85 |

**Table 6** continued

| | Micro-avg Scores | | | Macro-avg scores | | | Training Time (s) | Prediction Time (s) | Time Diff (%) |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | | | |
| C-RoBERTa-BASE | 0.95 | 0.917 | 0.933 | 0.948 | 0.901 | 0.924 | 79.08 | 2.84 | 118.46 |
| C-RoBERTa-MEDIUM | 0.982 | 0.917 | 0.948 | 0.974 | 0.917 | 0.945 | 81.5 | **2.75** | 111.54 |
| C-RoBERTa-LARGE | **0.997** | **0.949** | **0.972** | **0.974** | **0.947** | **0.96** | 79.08 | 2.81 | 116.15 |
| Dataset-6 | | | | | | | | | |
| RoBERTa (PLM-X2) | 0.82 | 0.77 | 0.794 | 0.93 | 0.91 | 0.92 | 81.2 | 3.01 | 109.03 |
| C-RoBERTa-BASE | 0.833 | 0.78 | 0.806 | 0.954 | 0.925 | 0.939 | 82.01 | 3.07 | 113.19 |
| C-RoBERTa-MEDIUM | 0.869 | 0.802 | 0.834 | 0.961 | 0.945 | 0.953 | 79.58 | **2.95** | 104.86 |
| C-RoBERTa-LARGE | **0.874** | **0.837** | **0.855** | **0.987** | **0.97** | **0.978** | 82.82 | 2.98 | 106.94 |
| Dataset-7 | Merged Dataset | | | | | | | | |
| RoBERTa (PLM-X2) | 0.97 | 0.91 | 0.939 | 0.87 | 0.82 | 0.838 | 597.65 | 25.98 | 0.15 |
| C-RoBERTa-BASE | 0.98 | 0.921 | 0.95 | 0.903 | 0.856 | 0.879 | 585.7 | **25.46** | -1.85 |
| C-RoBERTa-MEDIUM | 0.983 | 0.938 | 0.96 | 0.926 | 0.889 | 0.907 | 603.63 | 25.72 | -0.85 |
| C-RoBERTa-LARGE | **0.995** | **0.975** | **0.984** | **0.949** | **0.895** | **0.921** | 591.67 | 26.5 | 2.16 |

The BASE, MEDIUM, and LARGE models are related to the number of app reviews used in the pre-training. The Micro- and Macro-average scores for Precision (P), Recall (R) and F1 scores (F1) and the training and prediction times in seconds (s) are reported. The last column on the right represents the time differences in the percentage of PLMs compared to the Prior approaches. The best scores for each dataset are shown in bold. C-RoBERTa-LARGE has the highest score for all datasets

**Table 7** RQ3-1 results for datasets $D_1$–$D_4$: Comparison among the PLMs, Custom PLMs (i.e., trained on app reviews data), and the Prior approaches for binary classification

| Datasets & Approaches | P | R | F1 | Training Time (s) | Prediction Time (s) |
|---|---|---|---|---|---|
| Dataset-1 | Randomly Selected Class: Bug Report | | | | |
| AR-MINER | 0.37 | 0.36 | 0.36 | 260.92 | NA |
| SUR-MINER | 0.58 | 0.56 | 0.57 | 105.63 | 21.08 |
| Ensemble | 0.73 | 0.72 | 0.72 | 115.61 | 23.05 |
| DL+WE | 0.87 | 0.73 | 0.79 | **99.72** | **14.3** |
| BERT | 0.81 | 0.75 | 0.78 | 344.42 | 19.29 |
| ALBERT | 0.95 | 0.92 | 0.93 | 287.93 | 15.95 |
| RoBERTa | 0.91 | 0.83 | 0.87 | 358.78 | 19.87 |
| XLNET | 0.78 | 0.69 | 0.73 | 514.41 | 36.74 |
| C-ALBERT-BASE | 0.95 | 0.95 | 0.95 | 293.92 | 15.66 |
| C-ALBERT-MEDIUM | 0.96 | 0.95 | 0.95 | 281.89 | 15.52 |
| C-ALBERT-LARGE | 0.97 | 0.98 | 0.97 | 295.99 | 15.62 |
| C-RoBERTa-BASE | 0.98 | 0.95 | 0.96 | 361.64 | 19.55 |
| C-RoBERTa-MEDIUM | 0.98 | 0.97 | 0.97 | 359.84 | 20.17 |
| C-RoBERTa-LARGE | **1** | **0.99** | **0.99** | 366.32 | 19.37 |
| Dataset-2 | Randomly Selected Class: Inquiry | | | | |
| AR-MINER | 0.5 | 0.32 | 0.39 | 80.93 | NA |
| SUR-MINER | 0.6 | 0.56 | 0.58 | 27.04 | **3.88** |
| Ensemble | 0.81 | 0.7 | 0.75 | 29.78 | 4.99 |
| DL+WE | 0.94 | 0.86 | 0.9 | **26.34** | 5.29 |
| BERT | 0.8 | 0.7 | 0.75 | 184.58 | 10.32 |
| ALBERT | 0.98 | 0.91 | 0.94 | 120.29 | 6.69 |
| RoBERTa | 0.93 | 0.86 | 0.89 | 203.45 | 11.21 |
| XLNET | 0.87 | 0.78 | 0.82 | 345.92 | 24.66 |
| C-ALBERT-BASE | 0.91 | 0.94 | 0.92 | 117.77 | 6.8 |
| C-ALBERT-MEDIUM | 0.92 | 0.95 | 0.93 | 124.15 | 6.87 |
| C-ALBERT-LARGE | 0.92 | 0.98 | 0.95 | 120.65 | 6.87 |
| C-RoBERTa-BASE | **1** | 0.92 | 0.96 | 203.85 | 11.39 |
| C-RoBERTa-MEDIUM | 0.98 | 0.93 | 0.95 | 200.02 | 11.07 |
| C-RoBERTa-LARGE | **1** | **0.99** | **0.99** | 204.66 | 11.33 |

**Table 7** continued

| Datasets & Approaches | P | R | F1 | Training Time (s) | Prediction Time (s) |
|---|---|---|---|---|---|
| Dataset-3 | Randomly Selected Class: Reliability | | | | |
| AR-MINER | 0.44 | 0.36 | 0.4 | 74.77 | NA |
| SUR-MINER | 0.63 | 0.65 | 0.64 | **19.43** | **3.21** |
| Ensemble | 0.79 | 0.77 | 0.78 | 21.42 | 4.29 |
| DL+WE | 0.87 | 0.86 | 0.86 | 19.76 | 3.97 |
| BERT | 0.93 | 0.93 | 0.93 | 157.53 | 6.83 |
| ALBERT | 0.93 | 0.88 | 0.9 | 105.95 | 4.59 |
| RoBERTa | 0.95 | 0.89 | 0.92 | 175.64 | 7.66 |
| XLNET | 0.89 | 0.89 | 0.89 | 307.04 | 17.16 |
| C-ALBERT-BASE | 0.94 | 0.95 | 0.92 | 107.1 | 4.65 |
| C-ALBERT-MEDIUM | 0.93 | 0.94 | 0.95 | 107.02 | 4.49 |
| C-ALBERT-LARGE | 0.96 | 0.96 | 0.96 | 104.68 | 4.63 |
| C-RoBERTa-BASE | 0.98 | 0.97 | 0.96 | 173.37 | 7.72 |
| C-RoBERTa-MEDIUM | **1** | 0.97 | 0.98 | 177.93 | 7.5 |
| C-RoBERTa-LARGE | **1** | **1** | **1.0** | 177.22 | 7.46 |
| Dataset-4 | Randomly Selected Class: Feature Request | | | | |
| AR-MINER | 0.46 | 0.34 | 0.39 | 71.67 | NA |
| SUR-MINER | 0.64 | 0.59 | 0.61 | **11.76** | 2.35 |
| Ensemble | 0.85 | 0.75 | 0.8 | 13.54 | 2.7 |
| DL+WE | 0.88 | 0.78 | 0.83 | 12.23 | 2.05 |
| BERT | 0.86 | 0.8 | 0.83 | 75.84 | 2.9 |
| ALBERT | 0.86 | 0.83 | 0.84 | 53.93 | 2.06 |
| RoBERTa | 0.96 | 0.93 | 0.94 | 87.87 | 3.36 |
| XLNET | 0.79 | 0.78 | 0.78 | 148.91 | 7.14 |
| C-ALBERT-BASE | 0.94 | 0.94 | 0.9 | 54.15 | **2.02** |
| C-ALBERT-MEDIUM | 0.94 | 0.94 | 0.91 | 53.33 | 2.03 |
| C-ALBERT-LARGE | 0.96 | 0.97 | 0.91 | 52.38 | 2.07 |
| C-RoBERTa-BASE | 0.97 | 0.95 | 0.94 | 88.31 | 3.42 |
| C-RoBERTa-MEDIUM | **1** | 0.97 | 0.98 | 89.27 | 3.39 |
| C-RoBERTa-LARGE | **1** | **0.99** | **0.99** | 89 | 3.42 |

The Precision (P), Recall (R), F1 score (F1), and the training and prediction times in seconds (s) are reported. The best scores for each dataset are shown in bold. C-RoBERTa-LARGE scored highest, but, the model with the least prediction time varies as the dataset changes

**Table 8** RQ3-1 results for datasets $D_5$–$D_7$: Comparison among the PLMs, Custom PLMs (i.e., trained on app reviews data), and the Prior approaches for binary classification

| Datasets & Approaches | P | R | F1 | Training Time (s) | Prediction Time (s) |
|---|---|---|---|---|---|
| Dataset-5 | Randomly Selected Class: App Problem | | | | |
| AR-MINER | 0.54 | 0.35 | 0.42 | 56.96 | NA |
| SUR-MINER | 0.63 | 0.56 | 0.59 | **9.08** | 1.3 |
| Ensemble | 0.88 | 0.77 | 0.82 | 10.94 | 2.18 |
| DL+WE | 0.85 | 0.76 | 0.8 | 9.72 | 1.62 |
| BERT | 0.91 | 0.82 | 0.86 | 74.45 | 2.58 |
| ALBERT | 0.95 | 0.91 | 0.93 | 38.82 | 1.32 |
| RoBERTa | 0.97 | 0.92 | 0.94 | 80.53 | 2.78 |
| XLNET | 0.9 | 0.81 | 0.85 | 127.08 | 4.92 |
| C-ALBERT-BASE | 0.91 | 0.94 | 0.92 | 39.6 | **1.29** |
| C-ALBERT-MEDIUM | 0.93 | 0.97 | 0.95 | 37.82 | 1.3 |
| C-ALBERT-LARGE | 0.94 | 0.98 | 0.96 | 38.35 | 1.35 |
| C-RoBERTa-BASE | 0.98 | 0.92 | 0.95 | 79.48 | 2.83 |
| C-RoBERTa-MEDIUM | **1** | 0.93 | 0.96 | 81.34 | 2.74 |
| C-RoBERTa-LARGE | **1** | **0.97** | **0.98** | 79.48 | 2.82 |
| Dataset-6 | Randomly Selected Class: Bug Report | | | | |
| AR-MINER | 0.48 | 0.42 | 0.45 | 71.98 | NA |
| SUR-MINER | 0.59 | 0.53 | 0.56 | **10.02** | **1.44** |
| Ensemble | 0.76 | 0.75 | 0.75 | 11.92 | 1.72 |
| DL+WE | 0.78 | 0.77 | 0.77 | 10.32 | 1.47 |
| BERT | 0.81 | 0.75 | 0.78 | 75.13 | 2.76 |
| ALBERT | 0.82 | 0.79 | 0.8 | 48.79 | 1.82 |
| RoBERTa | 0.84 | 0.78 | 0.81 | 81.52 | 3.01 |
| XLNET | 0.75 | 0.67 | 0.71 | 142.72 | 6.24 |
| C-ALBERT-BASE | 0.79 | 0.81 | 0.8 | 49.96 | 1.83 |
| C-ALBERT-MEDIUM | 0.81 | 0.84 | 0.82 | 49.91 | 1.78 |
| C-ALBERT-LARGE | 0.82 | **0.86** | 0.84 | 49.51 | 1.82 |
| C-RoBERTa-BASE | 0.85 | 0.79 | 0.82 | 82.17 | 3.08 |
| C-RoBERTa-MEDIUM | 0.88 | 0.81 | 0.84 | 79.66 | 2.94 |
| C-RoBERTa-LARGE | **0.89** | 0.84 | **0.86** | 82.65 | 2.99 |

**Table 8** continued

| Datasets & Approaches | P | R | F1 | Training Time (s) | Prediction Time (s) |
|---|---|---|---|---|---|
| Dataset-7 | Merged Dataset; Randomly Selected Class: Aspect Evaluation | | | | |
| AR-MINER | 0.51 | 0.35 | 0.42 | 402.77 | NA |
| SUR-MINER | 0.68 | 0.56 | 0.61 | 157.01 | 31.25 |
| Ensemble | 0.78 | 0.71 | 0.74 | 177.6 | 29.45 |
| DL+WE | 0.86 | 0.79 | 0.82 | **156.13** | 26.04 |
| BERT | 0.93 | 0.86 | 0.89 | 573.75 | 25.04 |
| ALBERT | 0.97 | 0.87 | 0.92 | 462.49 | 20.05 |
| RoBERTa | **1.0** | 0.93 | 0.96 | 600.64 | 25.88 |
| XLNET | 0.94 | 0.88 | 0.91 | 834.45 | 48.98 |
| C-ALBERT-BASE | 0.94 | 0.98 | 0.96 | 463.4 | 20.11 |
| C-ALBERT-MEDIUM | 0.96 | 0.99 | 0.97 | 450.09 | 20.51 |
| C-ALBERT-LARGE | 0.96 | **1.0** | 0.98 | 467.98 | **19.79** |
| C-RoBERTa-BASE | 0.99 | 0.93 | 0.96 | 584.53 | 25.33 |
| C-RoBERTa-MEDIUM | 0.99 | 0.95 | 0.97 | 605.44 | 25.8 |
| C-RoBERTa-LARGE | **1.0** | 0.98 | **0.99** | 592.26 | 26.47 |

The Precision (P), Recall (R), F1 score (F1), and the training and prediction times in seconds (s) are reported. The best scores for each dataset are shown in bold. C-RoBERTa-LARGE scored highest, but, the model with the least prediction time varies as the dataset changes

**Table 9** RQ3-2 results: Comparison among the best PLMs (from RQ1 results), Custom PLMs (i.e., trained on app reviews data), and AR-Miner in zero-shot classification setting

| Dataset # | P | R | F1 | Prediction Time (s) |
|---|---|---|---|---|
| Dataset-1 | | | | |
| AR-MINER | 0.36 | 0.36 | 0.36 | N.A. |
| ALBERT-NLI | 0.47 | 0.42 | 0.44 | **15.35** |
| RoBERTa-NLI | 0.5 | 0.46 | 0.48 | 20.35 |
| C-ALBERT-LARGE-NLI | 0.5 | 0.47 | 0.48 | 16.7 |
| C-RoBERTa-LARGE-NLI | 0.59 | 0.53 | 0.56 | 20.66 |
| RoBERTa-LARGE-MNLI | **0.64** | **0.6** | **0.62** | 19.1 |
| Dataset-2 | | | | |
| AR-MINER | 0.49 | 0.32 | 0.39 | N.A. |
| ALBERT-NLI | 0.46 | 0.42 | 0.44 | 6.98 |
| RoBERTa-NLI | 0.5 | 0.47 | 0.48 | 10.85 |
| C-ALBERT-LARGE-NLI | 0.51 | 0.48 | 0.49 | **6.47** |
| C-RoBERTa-LARGE-NLI | 0.56 | 0.5 | 0.53 | 11.64 |
| RoBERTa-LARGE-MNLI | **0.64** | **0.58** | **0.61** | 10.8 |
| Dataset-3 | | | | |
| AR-MINER | 0.44 | 0.36 | 0.4 | N.A. |
| ALBERT-NLI | 0.43 | 0.4 | 0.41 | **4.37** |
| RoBERTa-NLI | 0.51 | 0.46 | 0.48 | 7.28 |
| C-ALBERT-LARGE-NLI | 0.55 | 0.51 | 0.53 | 4.72 |
| C-RoBERTa-LARGE-NLI | 0.61 | 0.57 | 0.59 | 7.95 |
| RoBERTa-LARGE-MNLI | **0.71** | **0.65** | **0.68** | 7.78 |
| Dataset-4 | | | | |
| AR-MINER | 0.45 | 0.33 | 0.38 | N.A. |
| ALBERT-NLI | 0.42 | 0.38 | 0.4 | **2.00** |
| RoBERTa-NLI | 0.51 | 0.47 | 0.49 | 3.22 |
| C-ALBERT-LARGE-NLI | 0.47 | 0.42 | 0.44 | 2.12 |
| C-RoBERTa-LARGE-NLI | 0.65 | 0.59 | 0.62 | 3.51 |
| RoBERTa-LARGE-MNLI | **0.69** | **0.64** | **0.66** | 3.21 |

**Table 9** continued

| Dataset # | P | R | F1 | Prediction Time (s) |
|---|---|---|---|---|
| Dataset-5 | | | | |
| AR-MINER | 0.53 | 0.35 | 0.42 | N.A. |
| ALBERT-NLI | 0.39 | 0.35 | 0.37 | **1.37** |
| RoBERTa-NLI | 0.53 | 0.49 | 0.51 | 2.91 |
| C-ALBERT-LARGE-NLI | 0.47 | 0.44 | 0.45 | **1.37** |
| C-RoBERTa-LARGE-NLI | 0.59 | 0.54 | 0.56 | 2.72 |
| RoBERTa-LARGE-MNLI | **0.74** | **0.67** | **0.70** | 2.9 |
| Dataset-6 | | | | |
| AR-MINER | 0.47 | 0.41 | 0.44 | N.A. |
| ALBERT-NLI | 0.4 | 0.36 | 0.38 | **1.73** |
| RoBERTa-NLI | 0.43 | 0.4 | 0.41 | 2.95 |
| C-ALBERT-LARGE-NLI | 0.43 | 0.41 | 0.42 | 1.88 |
| C-RoBERTa-LARGE-NLI | 0.52 | 0.48 | 0.5 | 3.13 |
| RoBERTa-LARGE-MNLI | **0.64** | **0.58** | **0.61** | 2.87 |
| Dataset-7 | | | | |
| AR-MINER | 0.49 | 0.34 | 0.4 | N.A. |
| ALBERT-NLI | 0.47 | 0.44 | 0.45 | **19.11** |
| RoBERTa-NLI | 0.58 | 0.53 | 0.55 | 24.94 |
| C-ALBERT-LARGE-NLI | 0.48 | 0.44 | 0.46 | 19.39 |
| C-RoBERTa-LARGE-NLI | 0.65 | 0.61 | 0.63 | 24.84 |
| RoBERTa-LARGE-MNLI | **0.74** | **0.68** | **0.71** | 27.28 |

The Precision (P), Recall (R), F1 score (F1), and prediction time in seconds (s) are reported. The best scores for each dataset are shown in bold. RoBERTa-LARGE-MNLI scored highest, and ALBERT-NLI has the lowest prediction time for most datasets

**Table 10** RQ3-3 results for Category classification task: Comparing the performance of PLMs and Custom PLMs (i.e., trained on app reviews data) in multi-task setting

| | Micro-Scores | | | Macro-Scores | | | Training Time (s) | Prediction Time (s) |
|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | | |
| Category Labeled Data | | | | | | | | |
| BERT | 0.813 | 0.759 | 0.791 | 0.752 | 0.714 | 0.732 | 287.16 | 9.988 |
| ALBERT | 0.883 | 0.865 | 0.86 | 0.842 | 0.785 | 0.823 | 230.095 | 12.006 |
| RoBERTa | 0.844 | 0.782 | 0.81 | 0.736 | 0.693 | 0.698 | 239.06 | 10.392 |
| XLNET | 0.798 | 0.717 | 0.753 | 0.822 | 0.721 | 0.779 | 418.48 | 19.692 |
| C-ALBERT-BASE | 0.888 | 0.85 | 0.872 | 0.853 | 0.785 | 0.807 | **139.437** | 12.126 |
| C-ALBERT-MEDIUM | 0.897 | 0.864 | 0.897 | 0.865 | 0.809 | 0.837 | 180.396 | 6.123 |
| C-ALBERT-LARGE | **0.933** | 0.89 | **0.917** | 0.859 | 0.804 | 0.831 | 281.634 | **5.943** |
| C-RoBERTa-BASE | 0.945 | 0.837 | 0.878 | 0.887 | 0.858 | 0.86 | 292.85 | 10.184 |
| C-RoBERTa-MEDIUM | 0.894 | 0.86 | 0.877 | **0.942** | 0.894 | 0.904 | 301.815 | 12.86 |
| C-RoBERTa-LARGE | 0.932 | **0.90** | 0.901 | 0.931 | **0.879** | **0.927** | 295.835 | 10.6 |

The Micro and Macro Precision (P), Recall (R), F1 score (F1), and the training and prediction times in seconds (s) are reported. The best scores for each dataset are shown in bold. The best scores belong to the LARGE Custom models

**Table 11** RQ3-3 results for Sentiment classification task: Comparing the performance of PLMs and Custom PLMs (i.e., trained on app reviews data) in multi-task setting

| | Micro-Scores | | | Macro-Scores | | | Training Time (s) | Prediction Time (s) |
|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | | |
| Sentiment Labeled Data | | | | | | | | |
| BERT | 0.852 | 0.769 | 0.804 | 0.759 | 0.748 | 0.739 | 689.184 | 32.461 |
| ALBERT | 0.886 | 0.889 | 0.877 | 0.848 | 0.806 | 0.822 | 552.228 | 26.013 |
| RoBERTa | 0.862 | 0.805 | 0.83 | 0.75 | 0.711 | 0.74 | 836.71 | 31.176 |
| XLNET | 0.808 | 0.737 | 0.787 | 0.849 | 0.754 | 0.794 | 1088.048 | 63.999 |
| C-ALBERT-BASE | 0.908 | 0.883 | 0.884 | 0.868 | 0.796 | 0.838 | 604.227 | 26.273 |
| C-ALBERT-MEDIUM | 0.913 | 0.878 | 0.89 | 0.879 | 0.829 | 0.845 | **541.188** | 26.533 |
| C-ALBERT-LARGE | 0.93 | 0.911 | 0.918 | 0.892 | 0.823 | 0.858 | 610.207 | **23.772** |
| C-RoBERTa-BASE | 0.958 | 0.861 | 0.904 | 0.913 | 0.858 | 0.903 | 644.27 | 33.098 |
| C-RoBERTa-MEDIUM | 0.934 | 0.864 | 0.913 | 0.955 | 0.905 | 0.929 | 784.719 | 28.292 |
| C-RoBERTa-LARGE | **0.964** | **0.934** | **0.93** | **0.972** | **0.907** | **0.935** | 828.338 | 31.800 |

The Micro and Macro Precision (P), Recall (R), F1 score (F1), and the training and prediction times in seconds (s) are reported. The best scores for each dataset are shown in bold. The best scores belong to the LARGE Custom models

**Table 12** RQ3-4 results for multi-resource setting: Comparing the performance of Prior approaches, PLMs, and Custom PLMs (i.e., trained on app reviews data) when the models are trained on app reviews and tested on Twitter data

| Datasets & Approaches | P | R | F1 | Prediction Time (s) |
|---|---|---|---|---|
| Dataset-2 | | | | |
| AR-Miner | 0.42 | 0.28 | 0.34 | N.A. |
| SUR-Miner | 0.61 | 0.56 | 0.58 | 39.22 |
| Ensemble | 0.78 | 0.72 | 0.75 | 38.79 |
| DL+WE | 0.85 | 0.78 | 0.81 | 27.79 |
| BERT | 0.57 | 0.50 | 0.53 | 17.49 |
| ALBERT | 0.69 | 0.65 | 0.67 | **9.99** |
| RoBERTa | 0.67 | 0.60 | 0.63 | 18.0 |
| XLNET | 0.64 | 0.56 | 0.60 | 44.42 |
| C-ALBERT-BASE | 0.75 | 0.67 | 0.71 | 11.54 |
| C-ALBERT-MEDIUM | 0.79 | 0.75 | 0.77 | 10.29 |
| C-ALBERT-LARGE | 0.84 | 0.76 | 0.80 | 12.35 |
| C-RoBERTa-BASE | 0.80 | 0.74 | 0.77 | 20.45 |
| C-RoBERTa-MEDIUM | 0.82 | 0.77 | 0.79 | 19.85 |
| C-RoBERTa-LARGE | **0.87** | **0.86** | **0.86** | 20.45 |

The Precision (P), Recall (R), F1 score (F1), and the prediction time in seconds (s) are reported. The best scores for each dataset are shown in bold. The best scores belong to the LARGE Custom model

**Table 13** RQ3-4 results for multi-resource setting: Comparing the performance of Prior approaches, PLMs, and Custom PLMs (i.e., trained on app reviews data) when the models are trained on Twitter data and are tested on app reviews

| Datasets & Approaches | P | R | F1 | Prediction Time (s) |
|---|---|---|---|---|
| Dataset-2 | | | | |
| AR-Miner | 0.36 | 0.24 | 0.29 | N.A |
| SUR-Miner | 0.66 | 0.59 | 0.62 | 25.1 |
| Ensemble | 0.83 | 0.76 | 0.79 | 23.66 |
| DL+WE | 0.89 | 0.83 | 0.86 | 16.4 |
| BERT | 0.62 | 0.55 | 0.58 | 9.62 |
| ALBERT | 0.73 | 0.72 | 0.72 | 6.29 |
| RoBERTa | 0.70 | 0.64 | 0.67 | 11.7 |
| XLNET | 0.69 | 0.62 | 0.65 | 28.43 |
| C-ALBERT-BASE | 0.79 | 0.70 | 0.74 | 6.81 |
| C-ALBERT-MEDIUM | 0.85 | 0.83 | 0.84 | 5.87 |
| C-ALBERT-LARGE | 0.93 | 0.82 | 0.87 | 6.79 |
| C-RoBERTa-BASE | 0.82 | 0.75 | 0.78 | 13.09 |
| C-RoBERTa-MEDIUM | 0.89 | 0.82 | 0.85 | 11.12 |
| C-RoBERTa-LARGE | **0.92** | **0.92** | **0.92** | 11.45 |

The Precision (P), Recall (R), F1 score (F1), and prediction time in seconds (s) are reported. The best scores for each dataset are shown in bold. The best scores belong to the LARGE Custom model

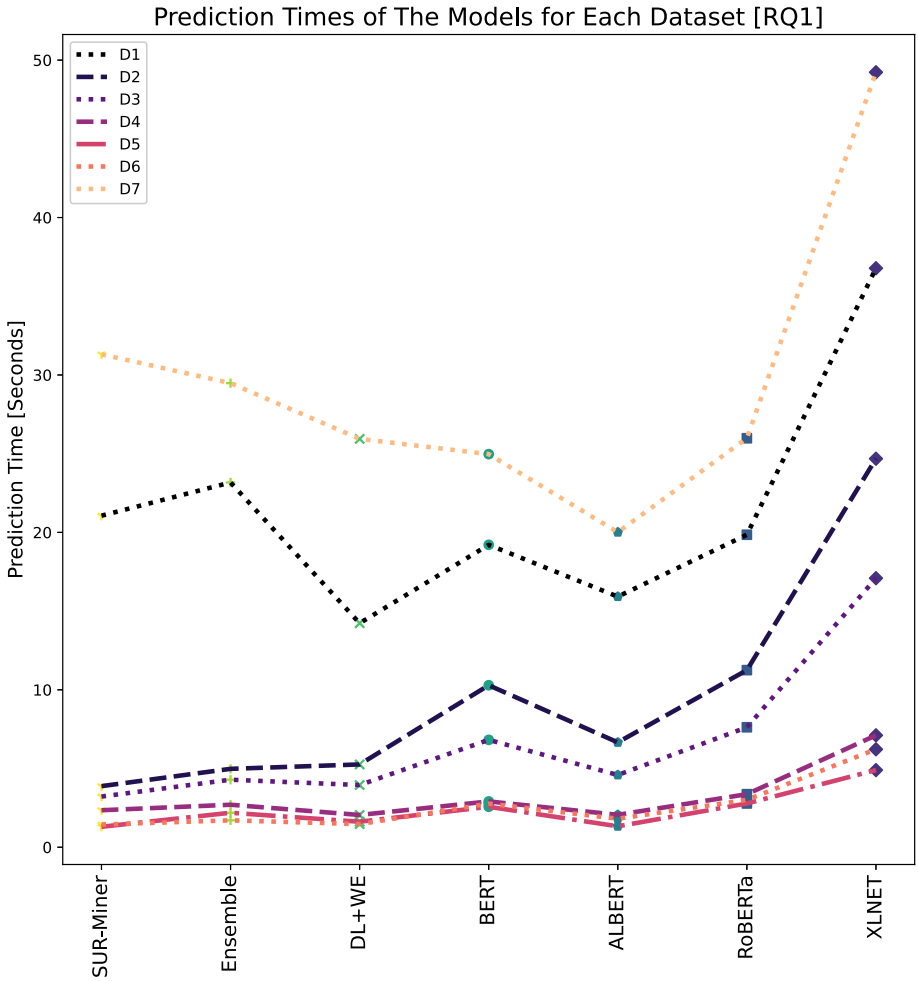


**Fig. 16** Prediction times of the Prior and PLMs for app issue classification (RQ1). Each line represents a dataset. The prediction times of $D_7$ are the highest, which can be related to the size of this dataset

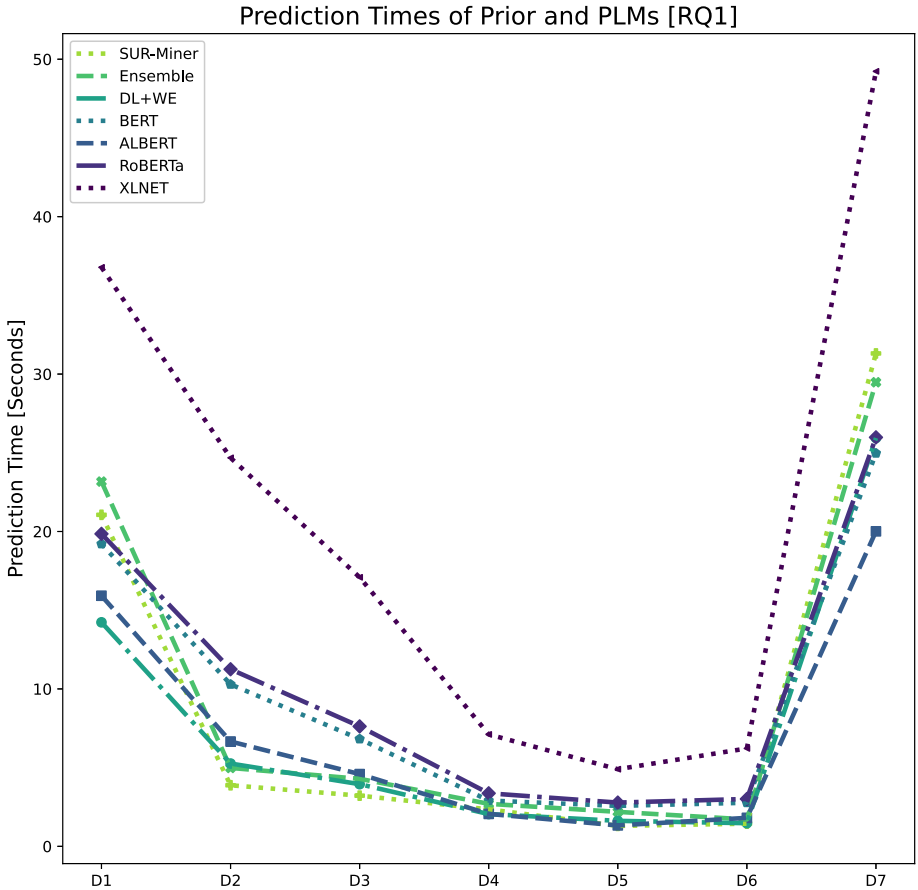


**Fig. 17** Prediction times of the Prior and PLMs for app issue classification (RQ1). Each model is shown in a colored line. The trend of the prediction time for the datasets is similar for all models, which can be related to the characteristics and size of each dataset. Among all PLMs, XLNET has the highest and ALBERT has the lowest prediction time

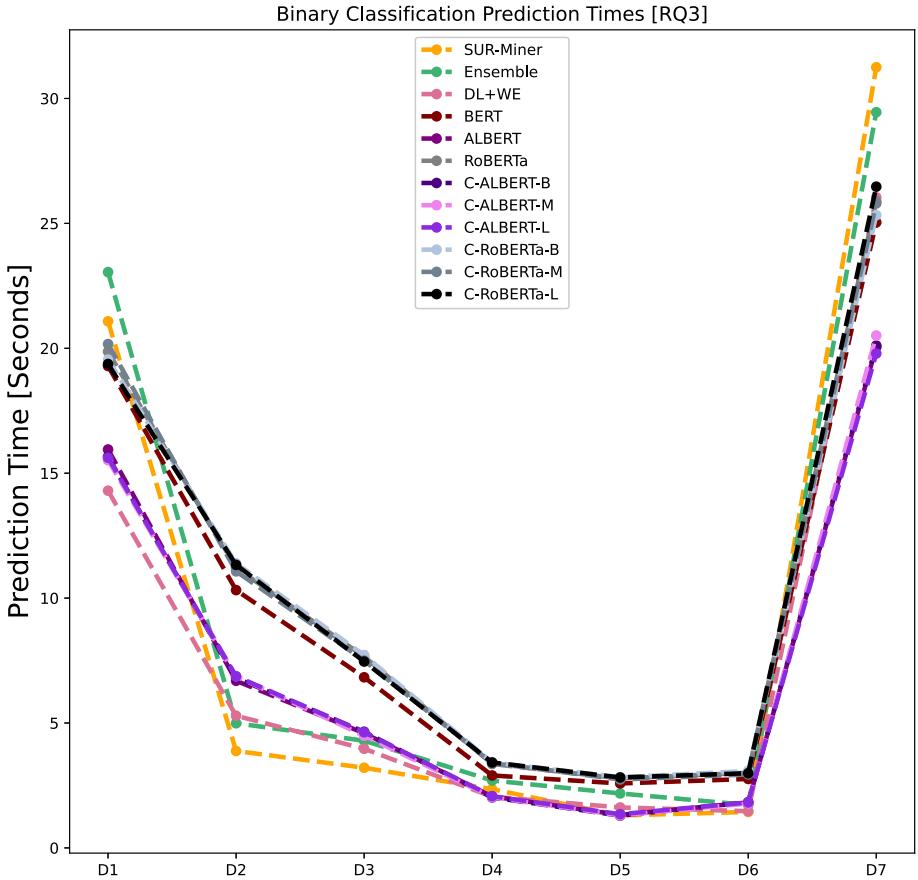


**Fig. 18** Prediction times of the models in binary classification setting (RQ3). The XLNET is not shown in the plot as its prediction times are higher than other models with a large gap. The Custom ALBERT-based models have the lowest scores for $D_4$, $D_5$, and $D_7$ datasets. RoBERTa-based models have higher prediction scores than ALBERT-based ones

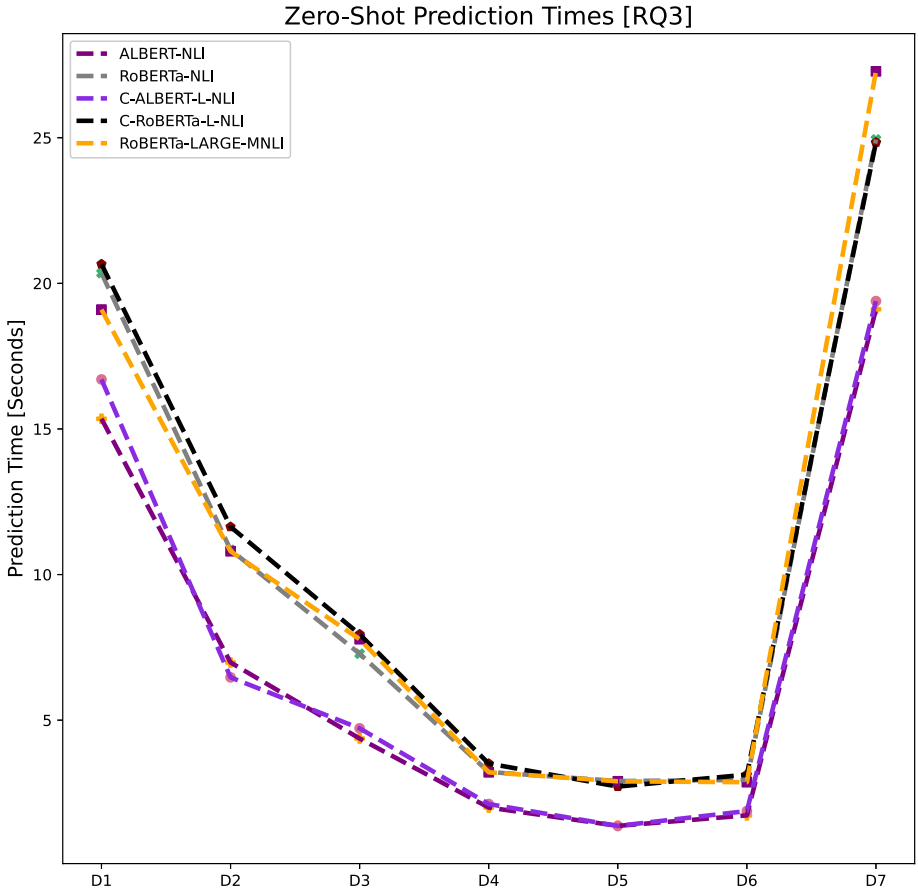


**Fig. 19** Prediction times of the models in zero-shot setting (RQ3). The ALBERT-based models have lower prediction times compared to the RoBERTa-based models

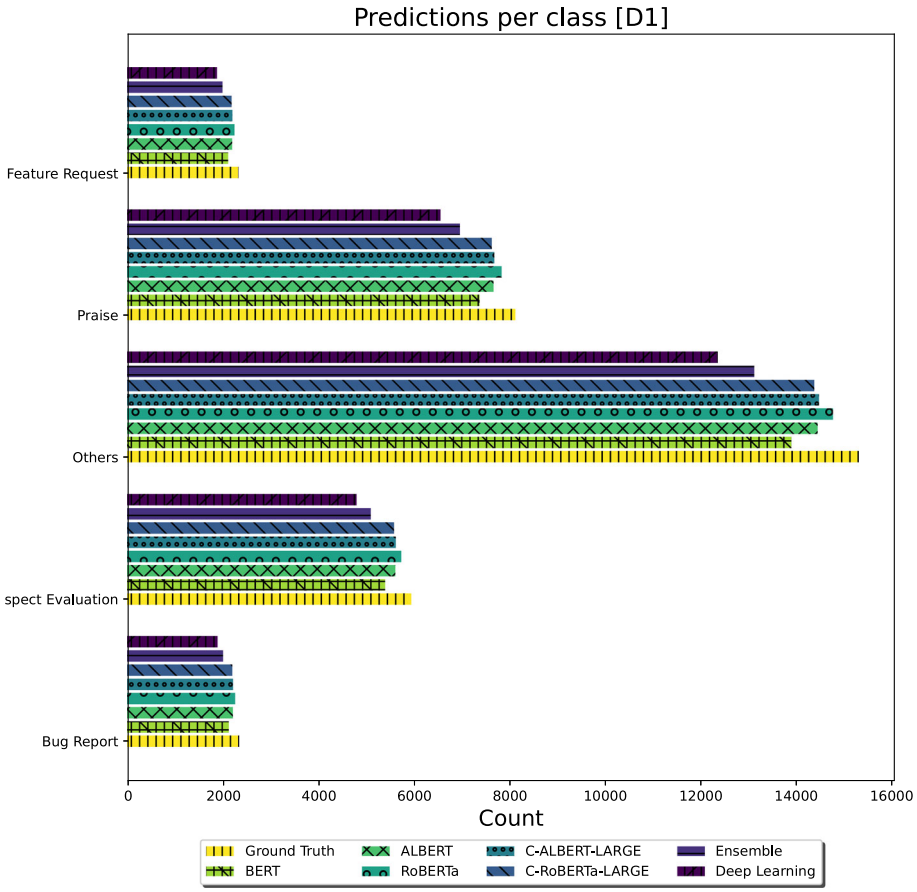


**Fig. 20** Prediction counts by each model divided by the class labels in dataset $D_1$. The counts are not normalized to show the differences among the classes

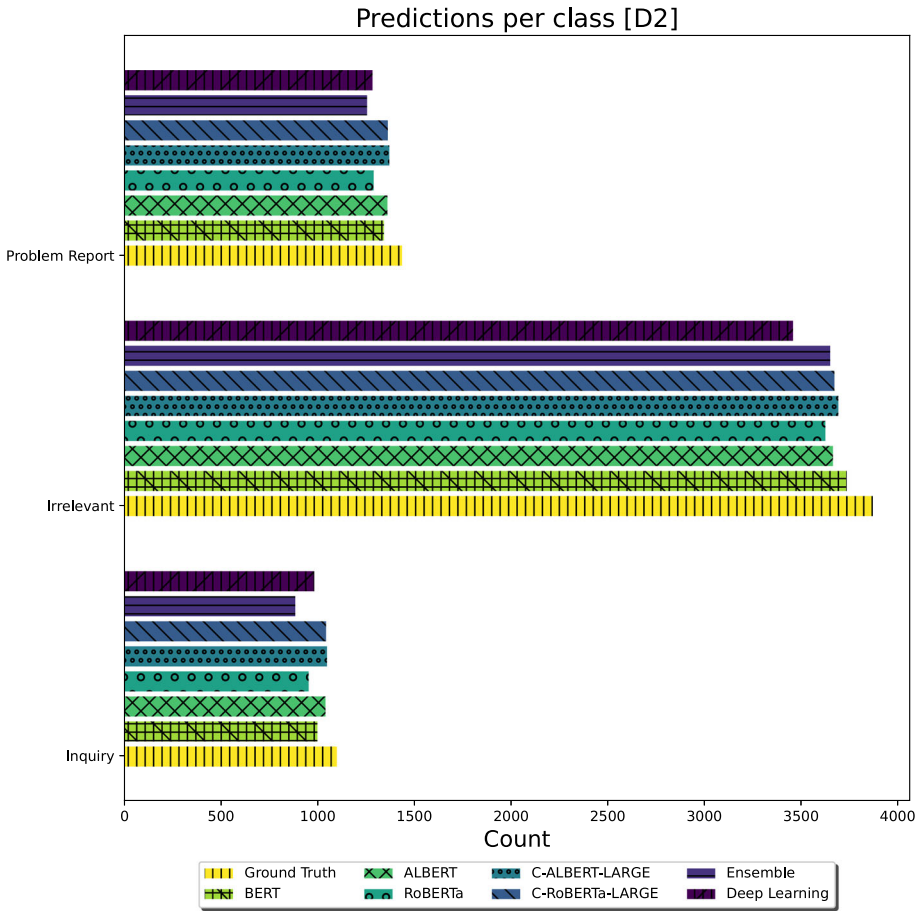


**Fig. 21** Prediction counts by each model divided by the class labels in dataset $D_2$. The counts are not normalized to show the differences among the classes

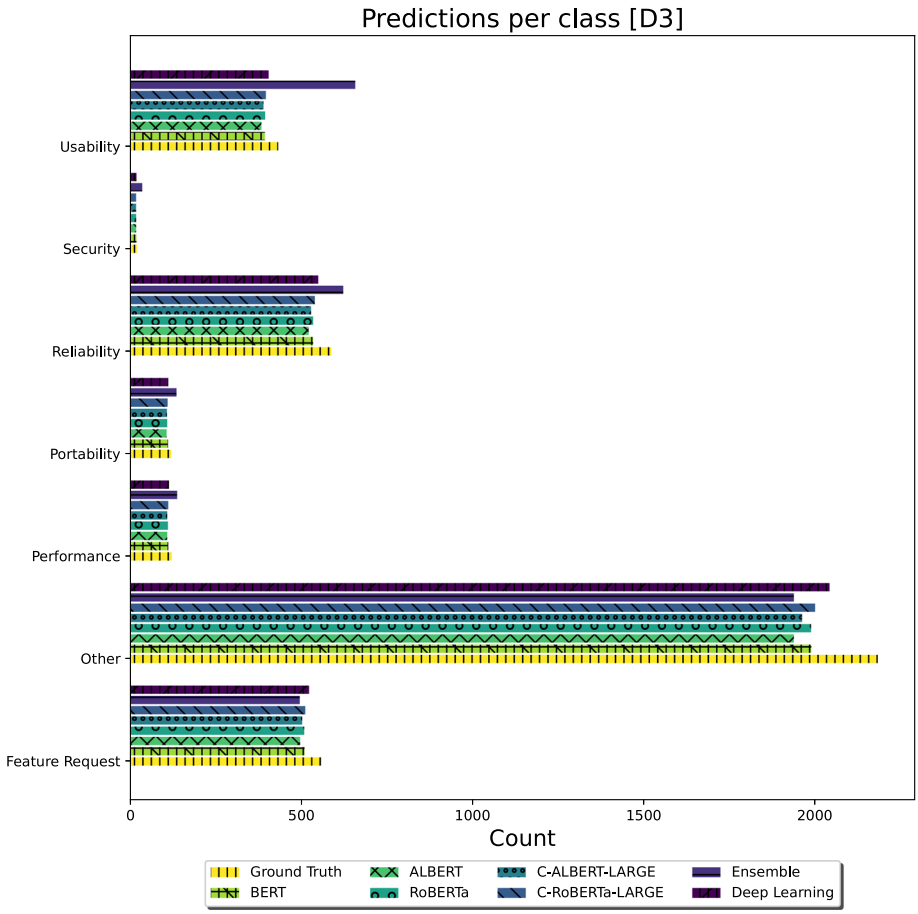


**Fig. 22** Prediction counts by each model divided by the class labels in dataset $D_3$. The counts are not normalized to show the differences among the classes

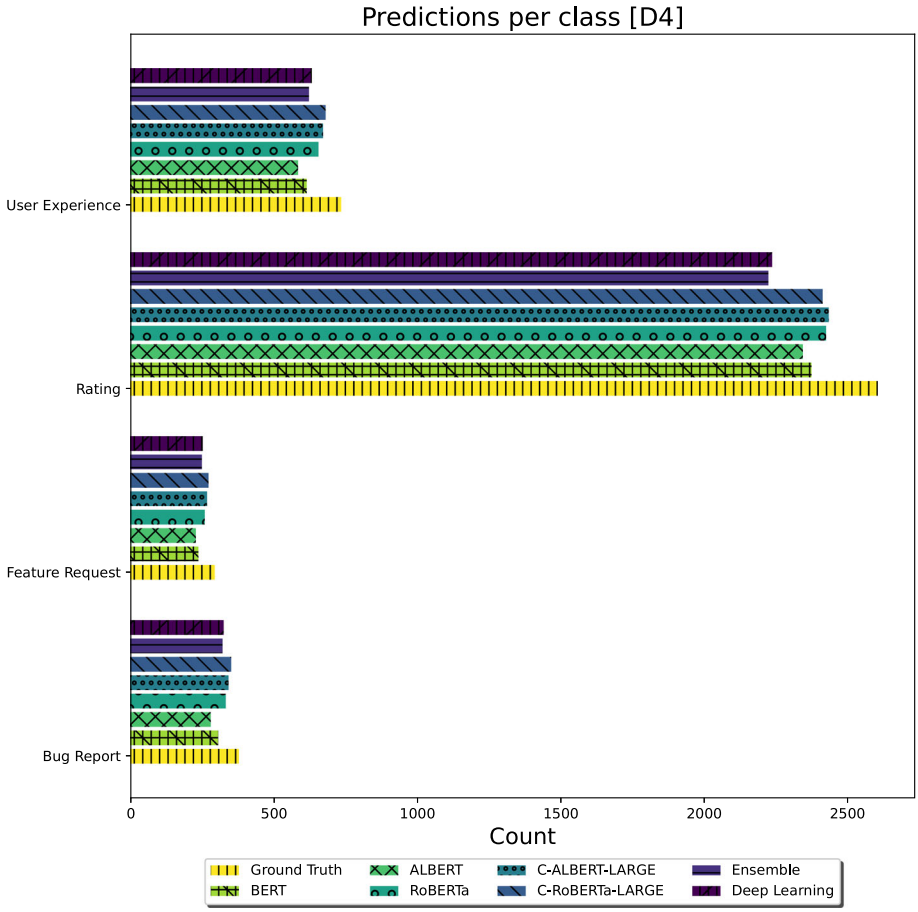


**Fig. 23** Prediction counts by each model divided by the class labels in dataset $D_4$. The counts are not normalized to show the differences among the classes

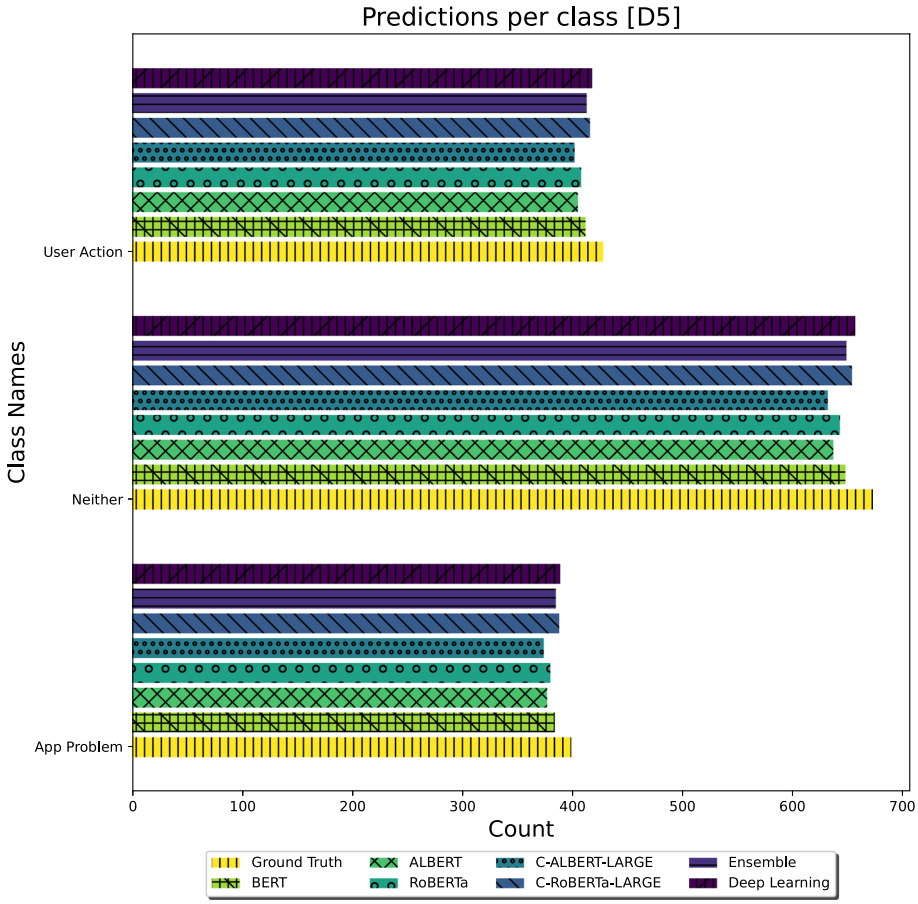


**Fig. 24** Prediction counts by each model divided by the class labels in dataset $D_5$. The counts are not normalized to show the differences among the classes

**Fig. 25** Prediction counts by each model divided by the class labels in dataset $D_6$. The counts are not normalized to show the differences among the classes

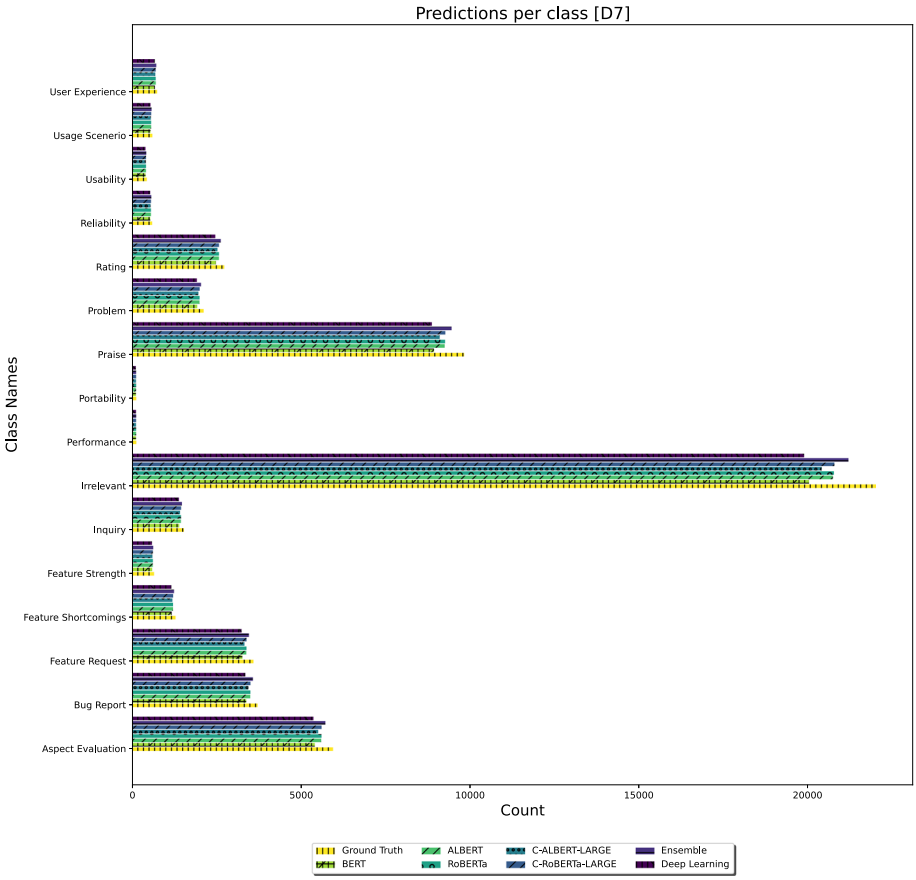


**Fig. 26** Prediction counts by each model divided by the class labels in dataset $D_7$. The counts are not normalized to show the differences among the classes

## References


Adhikari A, Ram A, Tang R, et al (2019) Docbert: Bert for document classification. arXiv preprint arXiv:1904.08398

Al-Hawari A, Najadat H, Shatnawi R (2021) Classification of application reviews into software maintenance tasks using data mining techniques. Softw Qual J 29:667–703

Ali M, Joorabchi ME, Mesbah A (2017) Same app, different app stores: A comparative study. In: 2017 IEEE/ACM 4th international conference on mobile software engineering and systems (MOBILESoft), pp 79–90

Allamanis M, Brockschmidt M, Khademi M (2018) Learning to represent programs with graphs. In: ICLR

Al-Subaihin AA, Sarro F, Black S et al (2021) App store effects on software engineering practices. IEEE Trans Softw Eng 47(2):300–319

Aralikatte R, Sridhara G, Gantayat N, et al (2018) Fault in your stars: An analysis of android app reviews. In: Proceedings of the ACM India joint international conference on data science and management of data. association for computing machinery, New York, NY, USA, CoDS-COMAD '18, pp 57–66

Araujo AF, Gôlo MP, Marcacini RM (2022) Opinion mining for app reviews: an analysis of textual representation and predictive models. Autom Softw Eng 29(1):1–30

Araujo A, Golo M, Viana B, et al (2020) From bag-of-words to pre-trained neural language models: Improving automatic classification of app reviews for requirements engineering. In: Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional, SBC. pp 378–389

Aslam N, Ramay WY, Xia K et al (2020) Convolutional neural network based classification of app reviews. IEEE Access 8:185619–185628

Bakiu E, Guzman E (2017) Which feature is unusable? detecting usability and user experience issues from user reviews. In: 2017 IEEE 25th International requirements engineering conference workshops (REW). pp 182–187

Bataa E, Wu J (2019) An investigation of transfer learning-based sentiment analysis in japanese. arXiv preprint arXiv:1905.09642

Bavota G, Linares-Vásquez M, Bernal-Cárdenas CE et al (2015) The impact of api change- and fault-proneness on the user ratings of android apps. IEEE Trans Softw Eng 41(4):384–407

Beltagy I, Lo K, Cohan A (2019) Scibert: A pretrained language model for scientific text. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP). Association for Computational Linguistics, Hong Kong, pp 3615–3620

Besmer AR, Watson J, Banks MS (2020) Investigating user perceptions of mobile app privacy: An analysis of user-submitted app reviews. Int J Inf Secur Priv (IJISP) 14(4):74–91

Biswas E, Karabulut ME, Pollock L, et al (2020) Achieving reliable sentiment analysis in the software engineering domain using bert. In: 2020 IEEE International conference on software maintenance and evolution (ICSME). pp 162–173

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3(Jan):993–1022

Cao Y, Fard FH (2021) Pre-trained neural language models for automatic mobile app user feedback answer generation. In: 2021 36th IEEE/ACM international conference on automated software engineering workshops (ASEW). pp 120–125

Chalkidis I, Fergadiotis M, Malakasiotis P, et al (2020) Legal-bert: The muppets straight out of law school. In: Findings of the association for computational linguistics: EMNLP 2020. Association for Computational Linguistics, Online, pp 2898–2904

Chang WC, Yu HF, Zhong K, et al (2019) X-bert: extreme multi-label text classification with using bidirectional encoder representations from transformers. arXiv preprint arXiv:1905.02331

Chen F, Fard FH, Lo D, et al (2022) On the transferability of pre-trained language models for low-resource programming languages. In: 2022 IEEE/ACM 30th international conference on program comprehension (ICPC). pp 401–412

Chen N, Lin J, Hoi SCH, et al (2014) Ar-miner: Mining informative reviews for developers from mobile app marketplace. In: Proceedings of the 36th international conference on software engineering. Association for Computing Machinery, New York, NY, USA, ICSE 2014, pp 767–778

Cimasa A, Corazza A, Coviello C, et al (2019) Word embeddings for comment coherence. In: 2019 45th Euromicro conference on software engineering and advanced applications (SEAA). pp 244–251

Ciurumelea A, Schaufelbühl A, Panichella S, et al (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). pp 91–102

Clinchant S, Jung KW, Nikoulina V (2019) On the use of bert for neural machine translation. arXiv preprint arXiv:1909.12744

Dabrowski J, Letier E, Perini A, et al (2019) Finding and analyzing app reviews related to specific features: A research preview. In: International working conference on requirements engineering: foundation for software quality, Springer, pp 183–189

Deocadez R, Harrison R, Rodriguez D (2017) Preliminary study on applying semi-supervised learning to app store analysis. In: Proceedings of the 21st international conference on evaluation and assessment in software engineering. Association for Computing Machinery, New York, NY, USA, EASE'17, pp 320–323

Devlin J, Chang MW, Lee K, et al (2019) Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, Minneapolis, Minnesota, pp 4171–4186

Dhinakaran VT, Pulle R, Ajmeri N, et al (2018) App review analysis via active learning: Reducing supervision effort without compromising classification accuracy. In: 2018 IEEE 26th International requirements engineering conference (RE), pp 170–181

Di Sorbo A, Panichella S, Alexandru CV, et al (2016) What would users change in my app? summarizing app reviews for recommending software changes. In: Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering. Association for Computing Machinery, New York, NY, USA, FSE 2016, pp 499–510

Edunov S, Baevski A, Auli M (2019) Pre-trained language model representations for language generation. In: Proceedings of the 2019 conference of the north american chapter of the association for computa-

tional linguistics: human language technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, Minneapolis, Minnesota, pp 4052–4059
Feng Z, Guo D, Tang D, et al (2020) Codebert: A pre-trained model for programming and natural languages. In: Findings of the association for computational linguistics: EMNLP 2020. Association for Computational Linguistics, Online, pp 1536–1547
Finkelstein A, Harman M, Jia Y, et al (2014) App store analysis: Mining app stores for relationships between customer, business and technical characteristics. RN 14(10):24
Forman G, Scholz M (2010) Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. SIGKDD Explor Newsl 12(1):49–57
Fu B, Lin J, Li L, et al (2013) Why people hate your app: Making sense of user feedback in a mobile app store. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery, New York, NY, USA, KDD '13, pp 1276–1284
Gao C, Zeng J, Lyu MR, et al (2018) Online app review analysis for identifying emerging issues. In: Proceedings of the 40th international conference on software engineering. Association for Computing Machinery, New York, NY, USA, ICSE '18, pp 48–58
Grano G, Ciurumelea A, Panichella S, et al (2018) Exploring the integration of user feedback in automated testing of android applications. In: 2018 IEEE 25th International conference on software analysis, evolution and reengineering (SANER). pp 72–83
Gu X, Kim S (2015) What parts of your apps are loved by users? (t). In: 2015 30th IEEE/ACM International conference on automated software engineering (ASE). pp 760–770
Guo B, Ouyang Y, Guo T et al (2019) Enhancing mobile app user understanding and marketing with heterogeneous crowdsourced data: A review. IEEE Access 7:68557–68571
Guo D, Ren S, Lu S, et al (2021) Graphcodebert: Pre-training code representations with data flow. In: Proceedings of the 2021 international conference on learning representations (ICLR)
Guo H, Singh MP (2020) Caspar: Extracting and synthesizing user stories of problems from app reviews. In: 2020 IEEE/ACM 42nd international conference on software engineering (ICSE). pp 628–640
Guzman E, Alkadhi R, Seyff N (2017) An exploratory study of twitter messages about software applications. Requir Eng 22(3):387–412
Guzman E, Alkadhi R, Seyff N (2016) A needle in a haystack: What do twitter users say about software? In: 2016 IEEE 24th international requirements engineering conference (RE), pp 96–105
Guzman E, El-Haliby M, Bruegge B (2015) Ensemble methods for app review classification: An approach for software evolution (n). In: 2015 30th IEEE/ACM International conference on automated software engineering (ASE). pp 771–776
Guzman E, Ibrahim M, Glinz M (2017b) A little bird told me: Mining tweets for requirements and software evolution. In: 2017 IEEE 25th International requirements engineering conference (RE). pp 11–20
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). pp 153–162
Hadi MA, Fard FH (2020) Aobtm: Adaptive online biterm topic modeling for version sensitive short-texts analysis. In: 2020 IEEE International conference on software maintenance and evolution (ICSME). pp 593–604
Hadi MA, Yusuf INB, Thung F, et al (2022) On the effectiveness of pretrained models for api learning. In: 2022 IEEE/ACM 30th International conference on program comprehension (ICPC). pp 309–320
Haering M, Stanik C, Maalej W (2021) Automatically matching bug reports with related app reviews. In: 2021 IEEE/ACM 43rd international conference on software engineering (ICSE). pp 970–981
Hakala K, Pyysalo S (2019) Biomedical named entity recognition with multilingual bert. In: Proceedings of the 5th workshop on BioNLP open shared tasks. pp 56–61
Harkous H, Peddinti ST, Khandelwal R, et al (2022) Hark: A deep learning system for navigating privacy feedback at scale. 2022 IEEE Symposium on Security and Privacy (SP)
He D, Hong K, Cheng Y et al (2019) Detecting promotion attacks in the app market using neural networks. IEEE Wirel Commun 26(4):110–116
He H, Ma Y (2013) Imbalanced learning: foundations, algorithms, and applications. NA
Hemmatian F, Sohrabi MK (2019) A survey on classification techniques for opinion mining and sentiment analysis. Artif Intell Rev 52(3):1495–1545
Henao PR, Fischbach J, Spies D, et al (2021) Transfer learning for mining feature requests and bug reports from tweets and app store reviews. In: 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW). pp 80–86
Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
Howard J, Ruder S (2018) Universal language model fine-tuning for text classification. In: Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Melbourne, Australia, pp 328–339

Huang Q, Xia X, Lo D et al (2020) Automating intention mining. IEEE Trans Softw Eng 46(10):1098–1119
Imamura K, Sumita E (2019) Recycling a pre-trained bert encoder for neural machine translation. In: Proceedings of the 3rd Workshop on Neural Generation and Translation, pp 23–31
Islam R, Islam R, Mazumder T (2010) Mobile application and its global impact. Int J Eng Technol 10(6):72–78
James G, Witten D, Hastie T et al (2013) An introduction to statistical learning, vol 112. Springer
Jha N, Mahmoud A (2019) Mining non-functional requirements from app store reviews. Empir Softw Eng 24(6):3659–3695
Johann T, Stanik C, Alizadeh B. AM, et al (2017) Safe: A simple approach for feature extraction from app descriptions and app reviews. In: 2017 IEEE 25th International requirements engineering conference (RE), pp 21–30
Joorabchi ME, Mesbah A, Kruchten P (2013) Real challenges in mobile app development. In: 2013 ACM / IEEE International symposium on empirical software engineering and measurement, pp 15–24
Ju Y, Zhao F, Chen S, et al (2019) Technical report on conversational question answering. arXiv preprint arXiv:1909.10772
Karimi A, Rossi L, Prati A (2021) Adversarial training for aspect-based sentiment analysis with bert. In: 2020 25th International conference on pattern recognition (ICPR), pp 8797–8803
Karmakar A, Robbes R (2021) What do pre-trained code models know about code? In: 2021 36th IEEE/ACM International conference on automated software engineering (ASE), pp 1332–1336
Kaur A, Kaur K (2022) Systematic literature review of mobile application development and testing effort estimation. J King Saud Univ Comput Inf Sci 34(2):1–15
Lan Z, Chen M, Goodman S, et al (2019) Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942
Li X, Bing L, Zhang W, et al (2019) Exploiting BERT for end-to-end aspect-based sentiment analysis. In: Proceedings of the 5th workshop on noisy user-generated text (W-NUT 2019). Association for Computational Linguistics, Hong Kong, China, pp 34–41
Li L, Ma R, Guo Q, et al (2020) Bert-attack: Adversarial attack against bert using bert. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP). Association for Computational Linguistics, Online, pp 6193–6202
Liu Y, Lapata M (2019) Text summarization with pretrained encoders. arXiv preprint arXiv:1908.08345
Liu Y, Ott M, Goyal N, et al (2019) Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692
Liu L, Ren X, Shang J, et al (2018) Efficient contextualized representation: Language model pruning for sequence labeling. In: Proceedings of the 2018 conference on empirical methods in natural language processing. Association for Computational Linguistics, Brussels, Belgium, pp 1215–1225
Liu W, Zhang G, Chen J, et al (2015) A measurement-based study on application popularity in android and ios app stores. In: Proceedings of the 2015 workshop on mobile big data. Association for Computing Machinery, New York, NY, USA, Mobidata '15, pp 13–18
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. Association for Computing Machinery, New York, NY, USA, EASE'17, pp 344–353
Luong K, Hadi M, Thung F, et al (2022) Arseek: Identifying api resource using code and discussion on stack overflow. In: 2022 IEEE/ACM 30th International conference on program comprehension (ICPC), pp 331–342
Maalej W, Kurtanović Z, Nabil H et al (2016) On the automatic classification of app reviews. Requir Eng 21(3):311–331
Maalej W, Nabil H (2015) Bug report, feature request, or simply praise? on automatically classifying app reviews. In: 2015 IEEE 23rd International requirements engineering conference (RE). pp 116–125
Martin W, Sarro F, Jia Y et al (2017) A survey of app store analysis for software engineering. IEEE Trans Softw Eng 43(9):817–847
Mekala RR, Irfan A, Groen EC, et al (2021) Classifying user requirements from online feedback in small dataset environments using deep learning. In: 2021 IEEE 29th International requirements engineering conference (RE), pp 139–149
Messaoud MB, Jenhani I, Jemaa NB et al (2019) A multi-label active learning approach for mobile app user review classification. International Conference on Knowledge Science. Springer, Engineering and Management, pp 805–816
Mikolov T, Chen K, Corrado G, et al (2013) Efficient estimation of word representations in vector space. In: ICLR Workshop
Mondal AS, Zhu Y, Bhagat KK, et al (2022) Analysing user reviews of interactive educational apps: a sentiment analysis approach. Interact Learn Environ 1–18

Nayebi M, Cho H, Ruhe G (2018) App store mining is not enough for app improvement. Empir Softw Eng 23(5):2764–2794

Nigam K, Lafferty J, McCallum A (1999) Using maximum entropy for text classification. IJCAI-99 workshop on machine learning for information filtering. Stockholom, Sweden, pp 61–67

Novielli N, Girardi D, Lanubile F (2018) A benchmark study on sentiment analysis for software engineering research. In: 2018 IEEE/ACM 15th International conference on mining software repositories (MSR), pp 364–375

Palomba F, Linares-Vásquez M, Bavota G, et al (2015) User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In: 2015 IEEE International conference on software maintenance and evolution (ICSME), pp 291–300

Palomba F, Salza P, Ciurumelea A, et al (2017) Recommending and localizing change requests for mobile apps based on user reviews. In: 2017 IEEE/ACM 39th International conference on software engineering (ICSE), pp 106–117

Panichella S, Di Sorbo A, Guzman E, et al (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), pp 281–290

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), pp 1532–1543

Peters ME, Ammar W, Bhagavatula C, et al (2017) Semi-supervised sequence tagging with bidirectional language models. In: Proceedings of the 55th annual meeting of the association for computational linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Vancouver, Canada, pp 1756–1765

Peters ME, Neumann M, Iyyer M, et al (2018) Deep contextualized word representations. In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, Volume 1 (Long Papers). Association for Computational Linguistics, New Orleans, Louisiana, pp 2227–2237

Qiao Z, Wang A, Abrahams A, et al (2020) Deep learning-based user feedback classification in mobile app reviews. In: Proceedings of the 2020 Pre-ICIS sigdsa symposium

Qiu X, Sun T, Xu Y et al (2020) Pre-trained models for natural language processing: A survey. Sci China Technol Sci 63:1869–1900

Qiu X, Sun T, Xu Y et al (2020) Pre-trained models for natural language processing: A survey. Sci China Technol Sci 63(10):1872–1897

Rajpurkar P, Zhang J, Lopyrev K, et al (2016) SQuAD: 100,000+ questions for machine comprehension of text. In: Proceedings of the 2016 conference on empirical methods in natural language processing. Association for Computational Linguistics, Austin, Texas, pp 2383–2392

Reddy S, Chen D, Manning CD (2019) Coqa: A conversational question answering challenge. Trans Assoc Comput Linguist 7:249–266

Reimers N, Schiller B, Beck T, et al (2019) Classification and clustering of arguments with contextualized word embeddings. In: Proceedings of the 57th annual meeting of the association for computational linguistics. Association for Computational Linguistics, Florence, Italy, pp 567–578

Ren Y, Zhang Y, Zhang M, et al (2016) Improving twitter sentiment classification using topic-enriched multi-prototype word embeddings. In: Thirtieth AAAI conference on artificial intelligence

Rietzler A, Stabinger S, Opitz P, et al (2019) Adapt or get left behind: Domain adaptation through bert language model finetuning for aspect-target sentiment classification. arXiv preprint arXiv:1908.11860

Robbes R, Janes A (2019) Leveraging small software engineering data sets with pre-trained neural networks. In: 2019 IEEE/ACM 41st International conference on software engineering: new ideas and emerging results (ICSE-NIER). IEEE, pp 29–32

Ruder S, Plank B (2017) Learning to select data for transfer learning with Bayesian optimization. In: Proceedings of the 2017 conference on empirical methods in natural language processing. Association for Computational Linguistics, Copenhagen, Denmark, pp 372–382

Rustam F, Mehmood A, Ahmad M et al (2020) Classification of shopify app user reviews using novel multi text features. IEEE Access 8:30234–30244

Santiago Walser R, De Jong A, Remus U (2022) The good, the bad, and the missing: Topic modeling analysis of user feedback on digital wellbeing features. In: Proceedings of the 55th Hawaii International Conference on System Sciences

Sarro F, Al-Subaihin AA, Harman M, et al (2015) Feature lifecycles as they spread, migrate, remain, and die in app stores. In: 2015 IEEE 23rd International requirements engineering conference (RE), pp 76–85

Scalabrino S, Bavota G, Russo B et al (2019) Listening to the crowd for the release planning of mobile apps. IEEE Trans Softw Eng 45(1):68–86

Shah FA, Sirts K, Pfahl D (2018) Simple app review classification with only lexical features. In: ICSOFT, pp 146–153

Shen V, jie Yu T, Thebaut S, et al (1985) Identifying error-prone software-an empirical study. IEEE Trans Softw Eng SE-11(4):317–324

Silva CC, Galster M, Gilson F (2021) Topic modeling in software engineering research. Empir Softw Eng 26(6):1–62

Stanik C, Haering M, Maalej W (2019) Classifying multilingual user feedback using traditional machine learning and deep learning. In: 2019 IEEE 27th International requirements engineering conference workshops (REW), pp 220–226

Subedi IM, Singh M, Ramasamy V, et al (2021) Application of back-translation: A transfer learning approach to identify ambiguous software requirements. In: Proceedings of the 2021 ACM Southeast Conference. Association for Computing Machinery, New York, NY, USA, ACM SE '21, pp 130–137

Sulistya A, Prana GAA, Sharma A et al (2020) Sieve: Helping developers sift wheat from chaff via cross-platform analysis. Empir Softw Eng 25(1):996–1030

Sun C, Huang L, Qiu X (2019) Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, Minneapolis, Minnesota, pp 380–385

Svyatkovskiy A, Deng SK, Fu S et al (2020) IntelliCode Compose: Code Generation Using Transformer. Association for Computing Machinery, New York, NY, USA, pp 1433–1443

Tang AK (2019) A systematic literature review and analysis on mobile apps in m-commerce: Implications for future research. Electron Commer Res Appl 37:100885

Tu M, Huang K, Wang G, et al (2020) Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In: Proceedings of the AAAI conference on artificial intelligence, pp 9073–9080

Van Nguyen T, Nguyen AT, Phan HD, et al (2017) Combining word2vec with revised vector space model for better code retrieval. In: 2017 IEEE/ACM 39th International conference on software engineering companion (ICSE-C). IEEE, pp 183–185

Vaswani A, Shazeer N, Parmar N, et al (2017) Attention is all you need. Advances in neural information processing systems 30

Villarroel L, Bavota G, Russo B, et al (2016) Release planning of mobile apps based on user reviews. In: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). pp 14–24

Von der Mosel J, Trautsch A, Herbold S (2022) On the validity of pre-trained transformers for natural language processing in the software engineering domain. IEEE Trans Softw Eng 1

Wada S, Takeda T, Manabe S, et al (2020) Pre-training technique to localize medical bert and enhance biomedical bert

Wallace E, Feng S, Kandpal N, et al (2019) Universal adversarial triggers for attacking and analyzing NLP. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP). Association for Computational Linguistics, Hong Kong, China, pp 2153–2162

Wang J, Wen R, Wu C et al (2020) Analyzing and Detecting Adversarial Spam on a Large-Scale Online APP Review System. Association for Computing Machinery, New York, NY, USA, pp 409–417

Wang C, Wang T, Liang P, et al (2019) Augmenting app review with app changelogs: An approach for app review classification. In: SEKE, pp 398–512

Wang C, Zhang F, Liang P, et al (2018) Can app changelogs improve requirements classification from app reviews? an exploratory study. In: Proceedings of the 12th ACM/IEEE International symposium on empirical software engineering and measurement. Association for Computing Machinery, New York, NY, USA, ESEM '18

Wan Y, Zhao W, Zhang H, et al (2022) What do they capture? a structural analysis of pre-trained language models for source code. In: Proceedings of the 44th international conference on software engineering. Association for Computing Machinery, New York, NY, USA, ICSE '22, pp 2377–2388

Wardhana JA, Sibaroni Y et al (2021) Aspect level sentiment analysis on zoom cloud meetings app review using lda. Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi) 5(4):631–638

Wolf T, Debut L, Sanh V, et al (2019) Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771

Wu X, Zhang T, Zang L, et al (2019) Mask and infill: Applying masked language model for sentiment transfer. In: Proceedings of the 28th international joint conference on artificial intelligence, IJCAI-19. International joint conferences on artificial intelligence organization, pp 5271–5277

Xu H, Liu B, Shu L, et al (2019) BERT post-training for review reading comprehension and aspect-based sentiment analysis. In: Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, Minneapolis, Minnesota, pp 2324–2335

Yang X, Macdonald C, Ounis I (2018) Using word embeddings in twitter election classification. Inf Retrieval J 21(2):183–207
Yang Z, Dai Z, Yang Y, et al (2019) Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems 32
Yang T, Gao C, Zang J, et al (2021) Tour: Dynamic topic and sentiment analysis of user reviews for assisting app release. In: Companion Proceedings of the Web Conference 2021. Association for Computing Machinery, New York, NY, USA, WWW '21, pp 708–712
Yang Z, Qi P, Zhang S, et al (2018b) HotpotQA: A dataset for diverse, explainable multi-hop question answering. In: Proceedings of the 2018 conference on empirical methods in natural language processing. Association for Computational Linguistics, Brussels, Belgium, pp 2369–2380
Yang C, Xu B, Khan JY, et al (2022) Aspect-based api review classification: How far can pre-trained transformer model go. In: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE Computer Society
Yatani K, Novati M, Trusty A, et al (2011) Review spotlight: A user interface for summarizing user-generated reviews using adjective-noun word pairs. In: Proceedings of the SIGCHI conference on human factors in computing systems. Association for Computing Machinery, New York, NY, USA, CHI '11, pp 1541–1550
Yin W, Hay J, Roth D (2019) Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP). Association for Computational Linguistics, Hong Kong, China, pp 3914–3923
Zhang X, Wei F, Zhou M (2019) HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In: Proceedings of the 57th annual meeting of the association for computational linguistics. Association for Computational Linguistics, Florence, Italy, pp 5059–5069
Zhang T, Xu B, Thung F, et al (2020) Sentiment analysis for software engineering: How far can pre-trained transformer models go? In: 2020 IEEE International conference on software maintenance and evolution (ICSME), pp 70–80
Zhang Z, Yang J, Zhao H (2021) Retrospective reader for machine reading comprehension. In: Proceedings of the AAAI conference on artificial intelligence, pp 14506–14514
Zhao L, Zhao A (2019) Sentiment analysis based requirement evolution prediction. Futur Internet 11(2):52
Zhao W, Guan Z, Chen L et al (2017) Weakly-supervised deep embedding for product review sentiment analysis. IEEE Trans Knowl Data Eng 30(1):185–197
Zhong M, Liu P, Chen Y, et al (2020) Extractive summarization as text matching. In: Proceedings of the 58th annual meeting of the association for computational linguistics. Association for Computational Linguistics, online, pp 6197–6208
Zhou Y, Su Y, Chen T et al (2021) User review-based change file localization for mobile applications. IEEE Trans Softw Eng 47(12):2755–2770

**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.

## Authors and Affiliations

**Mohammad A Hadi[1] · Fatemeh H. Fard[1]**

Mohammad A Hadi
mohammad.hadi@ubc.ca

[1] Department of Computer Science, University of British Columbia, Kelowna, Canada