Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Utilizing Creator Profiles for Predicting Valuable User Enhancement Reports

Feifei Niu[1] | Chuanyi Li*[1] | Heng Chen[1] | Jidong Ge[1] | Bin Luo[1] | Alexander Egyed[2]

[1]State Key Laboratory for Novel Software Technology, Software Institute, Nanjing University, Nanjing, China
[2]Institute for Software Systems Engineering, Johannes Kepler University, Linz, Austria

Correspondence
*Chuanyi Li, No.22, Hankou Road, Gulou District, Nanjing, China. Email: lcy@nju.edu.cn

Abstract

Users of software applications use Issue Tracking Systems (ITSs) to file enhancement reports, which leads to a large quantity of user requests. Indeed, these reports have become an important source for software requirements because they help to continuously improve software applications. Usually, developers and maintainers evaluate them and decide which user reports can be accepted. However, enhancement reports are continuously being raised one after another, which makes this process time-consuming and labor-intensive. Timely handling and implementation of these enhancement reports can effectively improve user satisfaction and product competitiveness. Thus, research has focused on automated methods for predicting which enhancement reports are likely to be approved, to maximum the value derived from user reports. However, reported results of existing approaches are typically not good enough for practical use. In this paper, we propose a novel creator-profile-based method to explore dependencies among enhancements to improve the prediction performance. Firstly, we define the concept of a creator profile, including a general method of how to generate creator profiles from the data set. Then we explain how to employ creator profiles to the problem of enhancement report approval prediction. Finally, we evaluate our approach on 40,551 enhancement reports collected from ITSs. The experimental results indicate that our proposed approach greatly improve on existing state of the art, especially in predicting approved reports. For cross-application prediction, the accuracy is 80.7%, while for non-cross-application prediction, the overall accuracy is 83.6%. That is, with the proposed approach, over 80% of user requests can be automatically identified for exacting valuable user requirements, which significantly reduces labor costs. Replication package is available at: https://anonymous.4open.science/r/approval_prediction-507E

KEYWORDS:
enhancement reports; approval prediction; creator profile; deep learning

1 | INTRODUCTION

During the evolution of software applications, users can propose issues to the application according to their use experience, which is vital to software improvement. Issue reports are proposed anytime and anywhere without the limitations of time, locations, or environments. This usually leads to a rapid growth of reports, especially whenever new versions are released. Typically, these reports reflect user wishes and are an important form of user participation during the software life cycle. According to the Standish Group's CHAOS Report 2015[1], user involvement is the third

element that should be focused on to improve project success. Indeed, implementation of users' enhancement reports can greatly improve the users' satisfaction and software competitiveness.

Normally, these user reports are either bug reports or enhancement reports. Different from bug reports, enhancement reports are suggestions for enhancing the system but they are often not essential for the software application. Usually, managers will not implement all the enhancement reports, but instead focus on those few ones deemed most valuable in terms of cost/benefit. According to Nizamani et al., less than 30% of the reports will be eventually approved [2]. Today, enhancement reports are manually analyzed by managers to decide whether to accept or not. Any surge in enhancement reports files increases handling cost (which is labor intensive) and may even lead to valuable enhancement reports not be processed in time or to be outright overlooked. Erne et al. revealed that user-driven innovation can only be significant when managers respond quickly to users' reports [3]. For example, comments by the managers and developers like "That's a good idea. But I don't have time to add this feature myself, but contributions are welcome." proof that many valuable suggestions are ignored due to lack of time. However, if there were an automated enhancement reports approval prediction (ERAP) method, a lot of time and cost could be saved for maintainers, and more enhancement reports could be handled to improve software competitiveness and users' satisfaction.

Current state of the art shows that several methods have been proposed for help in the approval prediction for enhancement reports. Nizamani et al., Umer et al, Nafees and Rehman employ traditional machine learning algorithms to implement classifiers for ERAP [2,4,5]. Arshad et al. and Cheng et al. trained deep learning classifiers for ERAP problem [6,7,8]. The most commonly used features for classification are semantic and syntactic information extracted from texts (including summary and description). In addition, sentiment analysis has been proven useful for ERAP [4]. In traditional text classification tasks, either project-specific keywords or non-project-specific keywords are indicative for classification [9]. However, since enhancement reports are usually fragmented, only concerning trivial aspect of the whole application [10], how can we judge whether to accept an enhancement report or not only through keywords? People do not repeatedly come up with features that already exist. Hence, informative words are often infrequent.

What we learn from state of the art is that analyzing the texts of enhancements reports is likely not sufficient for obtaining a good understanding of their value. Fortunately, more properties can be mined beyond the texts. Heppler et al. found that the probability of successful implementation of enhancement reports is twice as likely if the reports originate from developers rather than from common users [11]. The reason is that developers are more familiar with the application and are more skilled at proposing understandable reports. This makes them more likely to be approved. Therefore, the role of the creator (i.e., who filed the report) in the application may be as relevant for understanding ERAP as the text itself. Hence, this paper investigates this and propose the concept of a Creator Profile, which reflects on characteristic of the creator of enhancements reports. We combine creator profiles with deep learning based classifiers to solve the ERAP problem. We reuse the most common data set on ERAP, which is collected from Bugzilla. Each enhancement report is preprocessed with natural language processing techniques. Creator profiles for each report are calculated according to the definition. Finally, we use the resulting features for training and testing.

A creator profile consists of three parts: ID, Role and Frequency. More details will be explained in Section 3.2. The reason that creator profiles are useful can be explained from the following two aspects. Firstly, creators of different roles may have different approval rate according to Heppler's investigation [11]. So, distinguishing role of creators can be helpful. Another aspect is that creators with a higher frequency of producing enhancement reports may ultimately contribute more to the software application. In this way, their reports deserves higher attention.

In order to evaluate the proposed method, we reproduce the previous approaches, which are mainly based on Multinomial Naive Bayes (MNB) [2], Support Vector Machine (SVM) [5 4], and Convolution Neural Network (CNN) [7 6 8] classifiers. We employ 10-fold cross-validation including both cross-application and non-cross-application to report the results of different classifiers. Experimental results show the proposed creator profile greatly enhances the existing approaches in terms of key evaluation metrics. The contributions of this paper can be summarized as follows:

- We propose the concept of Creator Profile and demonstrate how to apply it to users' enhancement reports.
- We present a deep learning based approach to distinguish likely to-be-approved enhancement reports, which employs CNN classifier and combines both semantic features learned by BERT [12] embedding technique and creator information mined by creator profile.
- We reproduce existing approaches and make them publicly available, which are mainly based on SVM, MNB and CNN models for classification, as well as BOW (i.e., Bag of Words, such as Term Frequency) and Word2vec [13] to embed the texts. What's more, we also implement the common classification models: Multi-layer Perceptron (MLP, a feedforward artificial neural network), Long Short Term Memory (LSTM) [14], and language models GloVe [15] and BERT. We evaluate the different combinations of classifiers and language models on 40,551 enhancement reports from 10 open-source software applications, all of which were tracked and managed using Bugzilla.
- We evaluate the performance of the above approaches with and without creator profile. The cross-application and non-cross-application evaluation results of our approach are 80.7% and 83.6%, which outperforms the state-of-the-art approach. The results indicate that creator profile can improve the classifiers by an average of 4%.

The rest of this paper is organized as follows: Section 2 introduces related work from three aspects, namely, enhancement reports analysis, enhancement reports approval prediction, and deep learning in enhancement reports classification. Section 3 demonstrate the proposed approach. Section 4 describes and evaluates the proposed approach with 5 research questions. Section 5 explains threats to validity. Section 6 concludes the paper with an overview of future work.

2 | RELATED WORK

Recent years have witnessed the rapid growth of enhancement reports in software development. Many researches on enhancement reports analysis and management have emerged, especially approval prediction of enhancement reports.

2.1 | Enhancement Reports Analysis

Once proposed, enhancement reports are usually manually analyzed and managed by managers, which is time-consuming and labor-intensive. To cater to this, many automated analysis methods have been proposed. Shi et al. proposed a fuzzy rule and linguistic analysis-based method to structure enhancement reports. They generated a set of fuzzy rules to classify each sentence of enhancement reports into intent, explanation, benefit, etc [16]. Jayatilleke et al. introduced a framework for requirements change analysis, where they analyze the change using functions, identify the change difficulty and identify the dependencies using a matrix [17]. Morales-Ramire et al. exploited different properties of user feedback for requirements prioritization and proposed an automated approach for requirements prioritization [18]. Li et al. proposed a machine learning-based method to automatically classifying enhancement reports into capability, usability, security, etc [9]. To locate the source code that is closely related to feature requests, Zhang et al. use weight selection-based cosine similarity to retrieve the most relevant source code [19]. What's more, Xu et al. propose the MULAPI method to recommend API methods for new feature requests, based on the similarity between new feature requests and previous feature requests, as well as API documentations [20]. In general, research topics on enhancement reports mainly focus on understanding, prioritization, classification, code localization, as well as API recommendation. However, the basis of these analysis is that the enhancement report have been approved to be implemented. Otherwise, analyzing all the enhancement reports will cause a lot of redundant work. Therefore, ERAP is the foundation work which can avoid useless work by filtering out enhancement reports that will be rejected in the early stages.

2.2 | Enhancement Reports Approval Prediction

Nizamani et al. proposed a MNB based binary classifier to predict the approval status of enhancement reports from Bugzilla, where the textual reports were converted with vector space model (TF-IDF) [2]. Based on their research, Umer et al. enhanced the classifier by adding sentiment features [4]. They argued that emotional polarity is useful in approval prediction and calculated sentiment score with SENTI4SD [21]. Arshad et al. proposed to predict the resolution status of enhancement reports with the summary and description, as well as the sentiment labels [6,7]. They employed Word2Vec and deep learning classifier to learn the deep syntactical and semantic features in report texts. Nafees et al. proposed a Support Vector Machine-based (SVM) classifier to automatically classify enhancement reports into approved or rejected [5]. Cheng et al. proposed a CNN based approval prediction of enhancement reports with sentiment and deep representations [8]. Niu et al. proposed the concept of just-in-time feature request approval prediction and carried out preliminary experiments on a manually labelled data set [22]. However, existing researches mainly utilize text descriptions and sentiment analysis as features. They did not take into account other features that might be effective, especially, characteristic of creators.

2.3 | Deep Learning Based Enhancement Reports Classification

With the advance of deep learning techniques, a number of enhancement reports classification models have been proposed based on deep learning. Raul et al. proposed a CNN-supported model to classify functional requirements and non-functional requirements [23]. Shi et al. proposed FRMiner to detect feature requests from chat messages via a deep Siamese network and FRMiner outperformed four traditional text classifiers [24]. Liu et al. is the first that exploited CNN classifier into feature envy detection [25]. And their approach significantly outperformed other feature envy methods. Malik addressed the impact of review, reviewer, and project features on the review helpfulness prediction issue with machine learning techniques, where deep neural networks delivered the best result [26].

In this paper, we came up with creator profile, which can effectively boost existing approaches for ERAP task. We proposed a deep learning based classifier for ERAP task, which utilizes creator profile and is far beyond existing approaches.

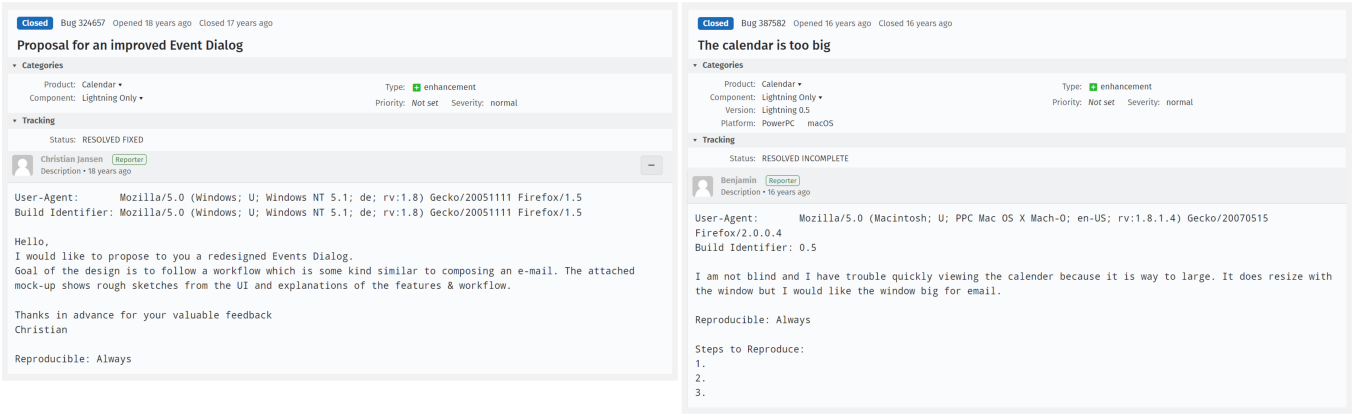

Figure 1 Example of two enhancement reports

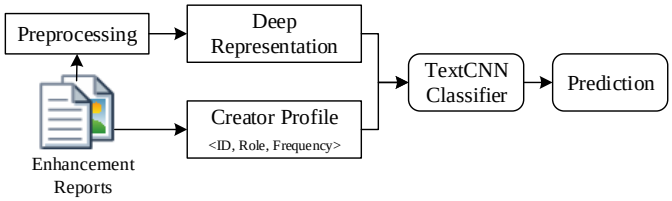

Figure 2 Overview of the proposed approach

3 | APPROACH

3.1 | Overview

Issue tracking systems (ITSs) bridge end users and software application developers by providing communication channels, where users file their enhancement reports and developers solicit users' suggestions for enhancing application. Enhancement reports largely consist of informal communication add-ons, e.g. greetings, refutations, negotiations, emoticons, etc. For example, Fig. 1 shows two enhancement reports of *Calendar*, which were resolved as FIXED [1] and INCOMPLETE[2]. They are both colloquial and somehow contains valueless information. Thus, requirements engineers have to go through these reports, discriminate users' intents and decide on whether to accept or reject. However, such process of understanding, classifying and clarifying end users' enhancement reports becomes a tedious and labor-intensive task, even for experienced requirements engineers [27].

Thus, in this paper, we propose an automatic approach to alleviate engineers' workload. With the historical data as training set, this approach can tell which enhancement report should be accepted, saving time on manual analysis. We define the problem as binary classification problem and utilize machine learning algorithms to predict whether an enhancement report will be approved or rejected. Fig. 2 provides an overview of our approach. Firstly, we obtain the enhancement reports from Bugzilla, which is one of the most popular ITSs. Then we preprocess all the enhancement reports with natural language processing techniques. We extract deep representations of preprocessed enhancement reports and creator profile information from each enhancement report as input of classifier. Finally we train deep learning model with the input features to predict the label of the test data. We also replicate the state-of-the-art models to compare with our approach in terms of evaluation metrics.

For users who plan to use our approach, our model can be easily integrated to ITSs, trained with resolved enhancement reports, from either cross-project or the same project. Whenever users file a new report, ITSs will be able to tell whether the report would be accepted or not immediately. For developers, the model will display valuable user enhancement reports to implement, while the manual selection process can be omitted, saving a lot of time and cost.

[1] https://bugzilla.mozilla.org/show_bug.cgi?id=324657
[2] https://bugzilla.mozilla.org/show_bug.cgi?id=387582

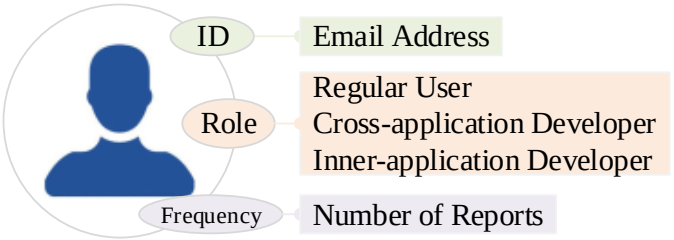

Figure 3 Creator Profile

3.2 | Creator Profile

The working assumption of this paper is that the role of the creator of enhancement reports is essential for the ERAP task. For example, it stands to reason that if a person is successful in creating a report that was the basis for a software change then this person is likely more successful with subsequent reports, than a novice. However, as far as we know, information about the creator is not being considered in existing work. Thus, in this paper, we propose the concept of a creator profile, which is used to describe the creator of enhancement reports. This work then uses the creator profile to improve the performance of existing classifiers in predicting software changes.

In this section, we introduce the *creator* profile as a 3-tuple, shown in Fig. 3, composing of ID, Role, and Frequency (Formula 1):

$$CP =< ID, Role, Frequency > \tag{1}$$

ID is used to uniquely identify each creator. In this paper, we use the email address of creators as their ID. We obtain the email address of enhancement report creators from the *creator* metadata, composing our user set.

Role stands for whether the creator is a developer or a user. Heppler et al. have confirmed that reports filed by regular users have less chances of being implemented than that of developers [11]. In order to distinguish developers from users, we use the *creator* and *assigned_to* metadata in the data set, where *creator* refers to the email of the reporter who filed the report and *assign_to* is the email of the developer who was assigned the report. If the creator of an enhancement report has also been assigned an enhancement report, it means that the creator is also a developer. To distinguish different levels of creators, we divide the creators into three levels: *regular user*, *cross-application developer* and *inner-application developer*. *Inner-application developer* means that the creator of the enhancement report is one of the developers of that same application. Here, the role of this creator will be assigned the value of 2. *Cross-application developer* refers that the creator is a developer of the other nine applications (there are 10 applications in the data set), but not involved in the same application the enhancement report is for. Here the creator's role for cross-application is assigned the value of 1. Finally, if the creator is neither a *cross-application developer* nor an *inner-application developer*, then he is a *regular user* and the role is assigned the value 0. Note that the *Role* of a creator is related to the application and is not static.

To identify whether the role is a regular user, inner-application developer or cross-application developer, we firstly obtain the set of email addresses from the *assign_to* list of all enhancement reports from the training set, which is Dev, where Dev_i stands for the set of email addresses in application i. Then, for each enhancement report in application i, we apply equation 2. Note that the applications do not provide their developer list. This method is curt and does not guarantee creators are annotated correctly. Errors may happen but it does provide a rough annotation and we will see later that this annotation is beneficial. In actual scenarios, project manages are aware of which individuals are developers, thus ensures more precise creator *Role*.

$$Role_i = \begin{cases} 0, & (ID \notin Dev) \\ 1, & (ID \in Dev) \wedge (ID \notin Dev_i) \\ 2, & (ID \in Dev_i) \end{cases} \tag{2}$$

Creator Frequency. Apart from creators' role, we also calculate the total submission number of each creator. The reasons can be described into two folds: On the one hand, we believe that application stakeholders tend to submit reports more frequently than regular users. The high frequency of submission may indicate that the creator is a developer. On the other hand, creators who make many reports tend to be more familiar to the application and have a better understanding of the application, or they know well how to make acceptable reports. Creators who only make only once or twice submissions are more likely to be rejected or ignored.

To calculate the submission frequency, firstly, we obtain the set of all creators. Then we count the frequency of occurrence for each creator in the data set, which is the creator's frequency.

For each application i, the CP of the creator can be represented by 3.

$$CP(i) =< ID, Role_i, Frequency > \tag{3}$$

3.3 | Preprocessing

We use the NLTK toolkits [28] to preprocess the enhancement reports, which include tokenization, spell checking, negative words replacement, POS tagging, inflection and lemmatization, stop words removing, and lowercase conversion. Before preprocessing, the summary and description of each enhancement is concatenated with spaces as the text feature of the report. First, punctuation (e.g., * and %) are replaced with spaces. Second, the text is tokenized and tagged with a POS tag. This converts texts into tokens with NLP tags. Third, we perform spell-check and negation on the tokenized text, which corrects the misspelled words and replace negation with antonyms. Fourth, stop-words (e.g., the, a, am, and are) are removed. Fifth, we perform inflection and lemmatization on all the filtered words. In this procedure, words are converted into singular form and their root words. POS tags can make the lemmatization and inflection results more accurate. Finally, we transform all the words into lower case and store the preprocessed reports. Previous research demonstrates that there is no need to remove stop words for deep learning classifiers [29]. So we remove stop words only for traditional machine learning models and retain them in deep learning models.

3.4 | Deep Representation

To represent enhancement reports with machine understandable language, we use distributed representation. In this paper, we employ word embedding techniques: BERT [12] to embed the sentences, which is pre-trained on a big corpus. Firstly, we tokenize each report. Then we use the pre-trained BERT [3] word embedding to convert the sentences into a matrix. The matrix is fed into classifiers to obtain the semantic features.

In addition to the distributed representation of the text, we also convert creator profile into numerical features. We encode *ID* of creator profile into a 300-dimensional vector. As for *Role*, there are three kinds of different creator roles: *regular user*, *cross-application developer*, *inner-application developer*. We encode the different creator roles with one-hot encoding, where each kind of role is represented with a 3-dimensional vector. Finally, the $Frequency$ is a one-dimensional vector represented by the frequency number.

3.5 | Deep Learning Classifier

Fig. 4 depicts our classifier. We employ CNN as our deep learning classifier, which is capable of learning semantic information with fast training speed.

The inputs of the classifier include two sections: text and creator profile (which includes three elements). Firstly, the pre-processed text is embedded with the BERT embedding layer. After that, the deep representation of the text is fed to 3 convolutional layers. The filters are 128 and the size of kernels are: 2, 3, 4. We use *relu* as activation function. Next, 3 max-pooling layers are employed to transfer the convolution outputs into vectors. The three max-pooling vectors are concatenated into one vector. Meanwhile, the creator profile contains 3 vectors, with *ID* embedded into a 300-dimensional vector, *Role* transferred into 3-dimensional vector and *Frequency* represented with 1-dimensional vector. We concatenate the four vectors and use fully connected layer to map the vectors into the predicted results. The activation function is *sigmoid*. The *adam* optimizer is employed and the learning rate is 0.0001. The loss function is *binary_crossentropy*.

4 | EVALUATION

In this section, first we introduce the data retrieval. Second, we explain the evaluation methodology and metrics. Third, we present the research questions. Finally, we display the experimental results and discuss the findings.

4.1 | Data Retrieval

ITSs are platforms dedicated for recording, tracking and managing issue reports [30]. Bugzilla [4] is one of the most popular ITSs, and is often used as the source of data set in ERAP researches. In this paper, we acquired users' enhancement reports from Bugzilla as our data set, through Bugzilla:REST API[5].

The metadata of enhancement reports on Bugzilla includes: *id*, *summary*, *type*, *resolution*, *creator*, *creation_time*, *assigned_to*, etc. The meaning of each metadata is explained as shown in Table 1. There are three kinds of issue reports, namely, *task*, *enhancement* and *defect*.Firstly, we only

[3]https://github.com/google-research/bert
[4]https://www.bugzilla.org/
[5]https://wiki.mozilla.org/Bugzilla:REST_API

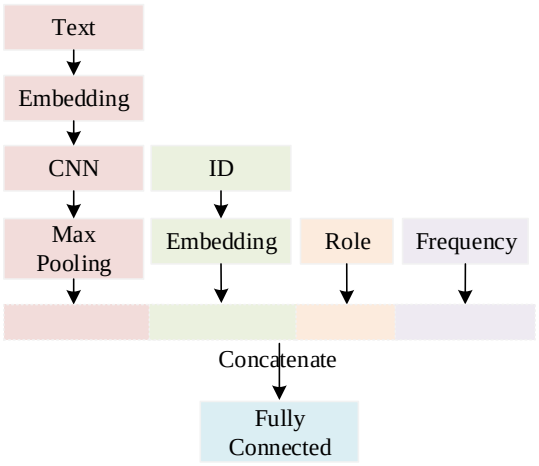

Figure 4 Framework of the approach

obtain enhancement reports [6] from Bugzilla. Next, detailed description of each enhancement report is extracted through report ID [7], which is the first comment proposed by the creator. We select the top ten most popular applications on Bugzilla and collect the enhancement reports proposed between 1997-09-10 and 2016-07-13. The metadata RESOLUTION indicates whether the report has been resolved or not. However, the resolution status "NULL" can not indicate the resolution status, so "NULL" reports are excluded from our data set.

Table 1 Meaning of the metadata.

Metadata	Description
id	The unique numeric ID of this report.
product	The name of the product this report is in.
severity	The current severity of the report.
resolution	The current resolution of the report (empty if it is open).
type	The type of it, including enhancement, bug, or task.
priority	The priority of the report.
summary	The summary of this report.
description	The actual text of the report.
creator	The name of the person filed this report (the reporter).
creation_time	When the report was created.
assigned_to	The name of the user to whom the report is assigned.

There are totally 40,551 enhancement reports in our data set, consisting of 10 applications. The resolution types include: INVALID, DUPLICATE, FIXED, WONTFIX, INCOMPLETE, WORKSFORME, EXPIRED, MOVED, and INACTIVE. Only FIXED reports are successfully resolved. To ensure a fair comparison with other state-of-the-art approaches, we adopted their data set partitioning strategy, where we labeled FIXED reports as APPROVED and others as REJECTED. The distribution of resolution status in different applications are as shown in Table 2.

[6] https://bugzilla.mozilla.org/rest/bug?type=enhancement
[7] https://bugzilla.mozilla.org/rest/bug/221/comment

Table 2 Distribution of Resolution Status in Different Applications.

Application	APPROVED	REJECTED	TOTAL
Bugzilla	2176	2422	4598
SeaMonkey	924	7111	8035
Core Graveyard	367	1202	1596
Core	2997	4790	7787
MailNews Core	407	1771	2178
Toolkit	422	1535	1597
Firefox	842	6594	7337
Thunderbird	451	3853	4304
Calendar	461	1139	1600
Camino Graveyard	347	839	1186
Total	9394	31157	40551

4.2 | Evaluation Methodology and Metrics

We use accuracy, precision, recall and F-measure metrics to evaluate the performance of the classifiers (see Formula 4, 5, 6 and 7). TP is the number of approved reports that are correctly predicted. FN is the number of approved reports that are wrongly predicted. FP is the number of rejected reports that are predicted as approved. TN is the number of rejected reports that are predicted correctly. Accuracy is the fraction of reports that are classified correctly. reports

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

To reduce the threats to validity, we use ten-folds cross validation strategy. There are two ways to divide data set into ten-folds: one way is cross-application [6,7,4,8] and the other way is non-cross-application [2,5]. To better compare with existing approaches, we adopt both data set split methods for evaluation. In the cross-application validation, we split the data set into ten folds according to the ten applications, with each application as one fold. In the non-cross-application validation, the enhancement reports of each application are equally divided into ten folds. What's more, the number of rejected reports in each fold is almost the same, so as approved reports.

In each experiment, we use one fold as test set and the other nine folds as training set, and calculate the overall accuracy, and precision, recall, F-measure of approved and rejected reports for each fold. Finally, we use the average of the ten folds as the experimental result.

4.3 | Research Questions

We evaluate our proposed approach by investigating the following research questions:

- **RQ1: How do different language models and classifiers perform on ERAP?**

  This RQ provides the comparison between different language models combined with different classifiers to investigate the performance. Then we choose the best language model and classifier as the start point of our method. Specifically, as far as we are concerned, there are 6 related researches in ERAP task. We reproduce these models and compare with the proposed approach.

- **RQ2: How does sentiment analysis impact the performance of the ERAP model?**

  This RQ investigates the influence of sentiment of the reports to the approval prediction. It evaluates whether positively written reports are more likely to be accepted by the managers.

- **RQ3: How does creator profile impact the performance of the ERAP model?**

  This RQ investigates the influence of creator profile to the approval prediction. It evaluates whether the creator profile helps to improve the performance of the approval prediction classifiers.

Table 3 Experimental Results on Different Classifiers–Cross-Application.

	Text							Text+Sentiment							Text+Creator Profile						
	Acc	Approved			Rejected			Acc	Approved			Rejected			Acc	Approved			Rejected		
		P	R	F	P	R	F		P	R	F	P	R	F		P	R	F	P	R	F
SVM-TF	24.1	57.4	0.1	0.2	24.1	**100**	37.5	24.1	56.6	0.1	0.1	24.1	**100**	37.5	51.4	24.8	**50.3**	31.6	76.4	51.2	60.6
SVM-TFIDF	35.5	74.1	16.5	20.9	28	84.7	36.6	24.6	51.8	0.7	1.3	24.1	99.6	37.5	52.2	24.5	49	31.5	76.4	51.6	61.2
MNB-TF	76.4	51	**45.5**	**45.1**	**83.5**	82.6	82.4	76.4	50.8	**45.8**	**45.3**	**83.6**	82.4	82.3	76.8	50.1	38.8	41.7	82.2	85.8	83.6
MNB-TFIDF	76.2	51.7	23.4	31.1	78.7	92.1	84.6	76.1	51.8	24.2	31.9	78.8	91.9	84.6	77.2	51.6	43.4	**45.5**	**83**	84.9	83.7
Arshd's	75.1	23.8	1.5	2.8	75.9	98.4	85.2	75.1	25.5	1.7	3	75.9	98.5	85.2	73.9	33.9	12.9	15.1	77.2	92.4	83.5
LSTM-word2vec	75.9	20.2	0.1	0.2	75.9	99.9	85.8	75	29.8	3.4	2.9	76.1	98.5	85.2	79.3	62.9	32.7	40.5	81.3	92.4	86.3
LSTM-GloVe	78.1	57.7	28.1	35.1	80.5	91.8	85.5	78.3	58.6	26.8	35	80.1	93.1	85.9	79.2	63.5	30.8	40.5	80.7	93.6	86.5
LSTM-BERT	78.7	58.9	31.5	38.8	81	92	86	78.5	58.9	30.2	37.7	80.8	92.3	86	79.6	64.8	33.3	42.4	81.4	93.1	86.7
CNN-word2vec	75.9	0	0	0	75.9	**100**	85.8	75.9	0	0	0	75.9	**100**	85.8	78	66.6	15.3	22.9	78.3	**97.3**	86.5
CNN-GloVe	76	48.3	3.8	6.9	76.3	98.8	85.6	75.7	42	4.1	7.4	76.3	98.3	85.5	78.1	56.8	30.8	39	80.5	91.9	85.6
CNN-BERT	**78.7**	**63.9**	26.2	35.7	80	94.5	**86.4**	**78.9**	**63.8**	27.8	37	80.4	94.2	**86.5**	**80.7**	**67.6**	34.2	43.6	82.1	92.9	**87.2**
TextGCN	78.2	55.2	22.8	30.7	80.2	93.9	85.9	77.6	60.9	17.6	25.2	79.8	95.5	85.9	78.7	62.4	25.7	34.9	81.2	93.8	86.2

Note: Acc represents Accuracy

- **RQ4: Does re-sampling help to improve the performance of the proposed approach? If yes, to what extent?**

  This RQ evaluates the impact of re-sampling. Since that the data set is uneven and the ratio between accepted and rejected reports is about 3:7. Therefore, we adopt different re-sampling strategies (i.e., over-sampling and under-sampling) to balance the data set and observe the change in evaluation metrics.

- **RQ5: What's the performance of our proposed approach on each fold?**

  This RQ shows the performance of the proposed method, the evaluation metrics on each fold as well as the time cost for training models.

4.4 | Experiments and Results

RQ1. How do different language models and classifiers perform on ERAP?

Method. To investigate the performance of different language models and classifiers, we reproduce the existing approaches on our data set, the approaches include: [2,4,8,7,5,6]. The classifiers include: SVM, MNB and CNN. The corresponding vector space models include TF and word embedding. What's more, we also evaluate MLP and LSTM classifier, as well as GloVe and BERT language models. We train and test the classifiers on both the cross-application and non-cross-application ten-fold cross-validation data set.

Results. The experimental results of cross-application and non-cross-application cross-validation are shown in the *Text* column in Table 3 and 4, respectively. For Non-cross-application validation, the accuracy is between 23.2% and 82.6%. The highest accuracy is obtained by CNN-BERT, which is 82.6%. The highest F1 measure for approved and rejected categories are 57.8% and 89.2%, which are obtained by MNB-TF and CNN-BERT, respectively. For cross-application validation, the accuracy is between 24.1% and 78.7%. The highest accuracy is 78.7%, gained by CNN-BERT. The best F1 measure for accepted reports is obtained by MNB-TF, that for rejected reports is obtained by CNN-BERT, which is 45.1%, 86.4%, respectively.

Discussion. As for different classification models, the experimental results show that deep classifiers generally perform better than traditional machine learning based classifiers. There is no significant difference between the accuracy of CNN, LSTM and TextGCN, among which, CNN is slightly better than others. The reason maybe that we use different filters to learn the semantic information of phrases of different lengths. In ERAP task, phrases may describe a kind of software feature, while software feature is more indicative than single words in ERAP. Meanwhile, in our context, the performance of SVM is unsatisfactory, where the accuracy is between 23% and 36%. The reason maybe that the data is uneven distributed, leading that it is difficult to find the hyperplane to distinguish between categories. This may indicate that SVM does not apply to our context. As for different text representations, for traditional machine learning classifiers, TF performs better on the MNB classifier, while TFIDF performs better on SVM classifier. Especially, MNB combined with TF gains the best recall and F1 on accepted reports and the best precision on rejected reports. Despite the uneven distribution of data, MNB is still able to identify minority class well. For deep learning based text representations, language models pre-trained based on large corpus, i.e., GloVe and BERT are much more effective in learning the semantic

Table 4 Experimental Results on Different Classifiers–Non-Cross-Application.

	Text							Text+Sentiment							Text+Creator Profile						
	Acc	Approved			Rejected			Acc	Approved			Rejected			Acc	Approved			Rejected		
		P	R	F	P	R	F		P	R	F	P	R	F		P	R	F	P	R	F
SVM-TF	23.2	66.2	0.1	0.2	23.2	**99.9**	37.6	23.2	67.2	0.1	0.2	23.2	**99.9**	37.6	54.4	21.8	39.9	27.2	76.5	58.8	65.7
SVM-TFIDF	33.3	81.2	16.9	23.6	25.1	87.6	38.1	23.3	**73.2**	0.2	0.4	23.2	99.8	37.6	49.5	21.9	45.3	29.1	73.5	50.7	59
MNB-TF	78.2	52.5	**64.3**	**57.8**	**88.4**	82.4	85.3	78.2	52.3	**64.6**	**57.8**	**88.5**	82.2	85.3	79.2	54.1	**66.9**	59.8	89.3	82.9	86
MNB-TFIDF	79.1	56.5	42.5	48.5	83.9	90.1	86.9	79	56.2	43.4	49	84	89.8	86.8	80.8	57.5	64.7	**60.9**	88.9	85.6	87.2
Arshd's	71.3	23	10.1	13.3	76.8	89.8	82.7	70	24	13.4	16.5	76.9	87	81.6	71.3	36.7	22.2	21.5	79.1	86	80.9
LSTM-word2vec	76.9	52.4	3.3	5.4	77.3	99.1	86.8	76.9	54.6	7.3	11.5	77.8	97.8	86.7	82.6	66.5	51.4	57.4	86.3	92	89.1
LSTM-GloVe	80.6	63.1	40.7	49.2	83.9	92.7	88	80.7	62.3	44.5	51.3	84.6	91.6	87.9	83.1	67.5	53.1	59.3	86.7	92.2	89.4
LSTM-BERT	81.1	64	43	51.3	84.4	92.6	88.3	81.1	63.1	45	52.3	84.7	92	88.2	82.2	66.1	47.5	55.2	85.4	92.6	88.9
CNN-word2vec	77.2	61.9	3.8	7.1	77.4	99.3	87	77.1	60.8	3.7	6.9	77.4	99.3	87	82.6	66.5	51.4	57.4	86.3	92	89.1
CNN-GloVe	81.4	64.9	46.2	53	85.1	92	88.3	81.1	63.1	45	52.3	84.7	92	88.2	83.3	**72.7**	45.8	55.7	85.3	**94.7**	89.7
CNN-BERT	**82.6**	**68.6**	46.4	55.3	85.3	93.6	**89.2**	**82.6**	68.9	45.9	55	85.2	93.7	**89.2**	**83.6**	70.5	50.5	58.8	**86.3**	93.6	**89.8**
TextGCN	81.6	66.6	41.4	51	84.1	93.7	88.7	81.5	67.7	38.6	49.1	83.6	94.4	88.7	82.1	66	45.6	53.8	85	92.9	88.8

Note: Acc represents Accuracy

information in texts than word2vec. Specifically, word2vec based models obtain very low recall and F1 score on minority class, i.e., approved. The model predicts almost all the test data as rejected.

As for the comparison of different combinations of classifiers and text representation techniques, CNN combined with BERT have the best performance. Arshad's model uses word2vec as word embedding technique, which is unable to learn the semantic information well. Thus the prediction result is not as accurate as CNN with BERT or GloVe.

RQ2. How does sentiment analysis impact the performance of the ERAP model?

Method. To evaluate the influence of sentiment analysis on the prediction models, we input the sentiment labels with other text features into the classification models in RQ1.

Just as previous researches did, we employ the SENTI4SD to perform sentiment analysis on the enhancement reports. We use the SENTI4SD model to predict the sentiment label for each enhancement report, which predicts each report as one of *Positive*, *Negative* and *Neutral*. After sentiment analysis, Arshad et al. embedded the emotional labels and then input to the convolutional layer. But in our experiments, we convert the emotional labels into one-dimensional vector with one-hot encoding. After that, we concatenate the one-dimensional vector with the TF or TFIDF vector to input into the traditional machine learning classifier, or directly concatenate the vector with the feature vectors learned from deep learning models and then output the predicted results after fully connected layers.

Results. The results of cross-application and non-cross-application validation are shown in the *Text+Sentiment* column in Table 3 and Table 4. *Text+Sentiment* stands for that the inputs include both text and sentiment analysis result. For cross-application prediction, the overall accuracy is between 24.1% and 78.9%. The F1 scores for accepted and rejected reports are between 0% and 45.3%, 37.5% and 86.5%, respectively. The combination of CNN and BERT achieves best classification accuracy, which is 78.9%. As for non-cross-application prediction, the overall accuracy is between 23.2% and 82.6%, where CNN combined with BERT achieves the best accuracy score and SVM with TF is the worst. The F1 scores for accepted and rejected reports are between 0.2% and 57.8%, 37.2% and 89.2%.

When compare the *Text* and *Text+Sentiment* columns in the two tables, there is no significant difference between the models with and without sentiment analysis as input in terms of the evaluation metrics. The results show that adding sentiment analysis results do not improve the performance of different models in our context. There is no obvious improvement in the overall accuracy, precision, recall and F1 score. We can conclude that sentiment analysis is not an indicative feature in our context.

Discussion. In order to analyze the reason why sentiment analysis does not work, we list the distribution of sentiment analysis results in the two categories, which is shown in Table 5. Among all the reports, the proportions of neutral, positive and negative are 66.7%, 18.6% and 14.7%. In the accepted reports, 18.1% of the reports are predicted as positive, while in the rejected reports, the proportion of positive reports are 18.8%. In our context, there is no significant difference in the distribution of sentiment analysis results on the two categories. The statement that positive or negative reports are more likely to be accepted is not robust. The reason that we guess may be due to the big discrepancy between our data set and previous data set. Although we adopt the same way to obtain our data set, the data on bugzilla is dynamically managed. So there can be big difference between our data set and that from previous studies. Therefore, we reach the conclusion that sentiment analysis is not a discriminate feature for ERAP in our context. Therefore, in our subsequent experiments, we did not employ sentiment analysis results.

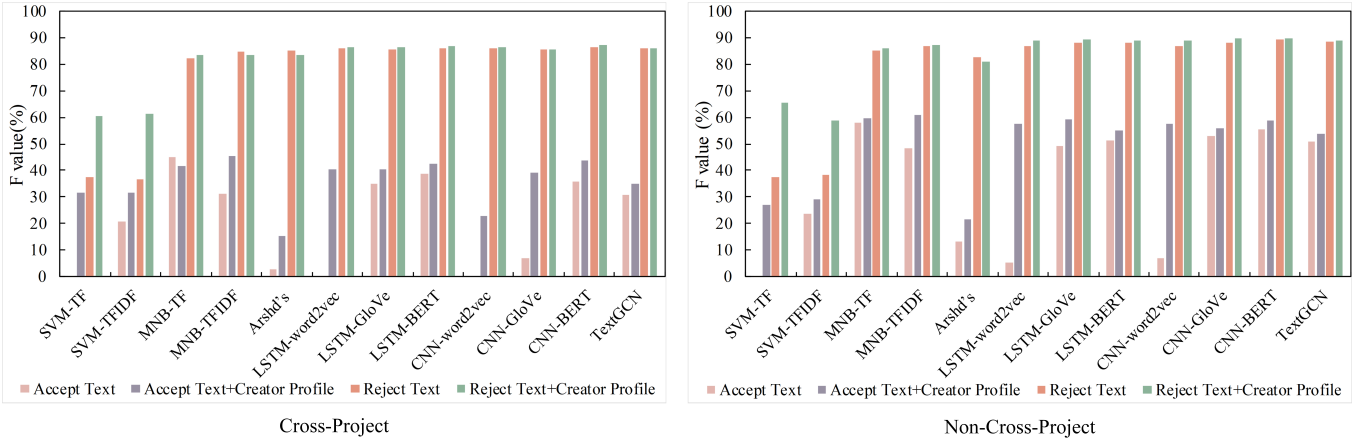

Figure 5 Improvement of F value with utilizing creator profile

Table 5 Sentiment Result in Different Classes.

	Approval	Rejected	Total
Positive	1701	5853	7554
Negative	1371	4596	5967
Neutral	6322	20708	27030

RQ3. How does creator profile impact the performance of the ERAP model?

Method. To evaluate the performance of creator profile, we use it as part of the inputs to the classifiers in RQ1. Firstly, we vectorize the creator profile according to Section 3.2. Then we concatenate the creator profile vector, including ID, Role and Frequency, with the vectorized features of the original text as input to the classifiers. Specifically, we concatenate the creator profile vectors with the TF or TFIDF vector for the traditional machine learning classifiers. We concatenate the creator profile vectors to the vectors generated by the convolutional layer or LSTM layer in the deep learning classifiers. Then, several fully connected layers are employed to output the predicted results.

Results. The *Text+Creator Profile* column in Table 3 and Table 4 show the performance of adding creator profile in different classifiers under different cross-validation data set division methods. The overall accuracy is between 51.4% and 80.7%, 54.4% and 83.6%, for cross-application and non-cross-application, respectively. For cross-application evaluation, the best F1 score for approved and rejected reports are 45.5% and 87.2%, respectively, which is achieved by MNB-TFIDF and CNN-BERT, respectively. For non-cross-application evaluation, the best F1 score for approved and rejected reports are 58.8% and 89.8%, which is both achieved by CNN-BERT.

Comparing the *Text* column and *Text+Creator Profile* column in the same row, we can notice a substantial improvement in the performance of the models in terms of the evaluation metrics when adding creator profile as model inputs. For SVM, the improvement in accuracy is up to 130%. For deep learning based classifiers, creator profile can improve the average accuracy by 1.65% and 2.5% for cross-application and non-cross-application cross-validation, respectively. Adding creator profile can effectively improve the F1 measure of both categories, especially the accepted reports, which is minority category. Due to the uneven distribution of categories, all the classifiers can not recognize accepted reports well, labeling almost all reports as rejected, which means most of the valuable enhancement reports are neglected by classifiers. But with creator profile, the precision, recall and F1 score of accepted reports are greatly improved, while that of the rejected reports stay almost unchanged. Fig 5 presents the improvement of F score with using creator profile in all models, since F score is the harmonic mean of precision and recall values. The F score of accepted reports shows significant improvement without reducing F score in rejected reports. In this way, all the classifiers become more reliable for developers to identify valuable user requests. Specifically, the creator profile can improve the accuracy of the state-of-the-art method by 4%.

Discussion. The experimental results indicate that creator profile is an encouraging feature for ERAP. The reasons can be explained from three aspects: ID, Role and Frequency. Firstly, table 6 shows the distribution of different creator roles in different categories. The acceptance rate of

Table 6 Creator Role Distribution in Different Classes.

Role	Approval	Rejected	Total
Regular User (0)	2853	24842	27695
Cross-application Developer (1)	487	1483	1970
Inner-application Developer (2)	6054	4832	10886

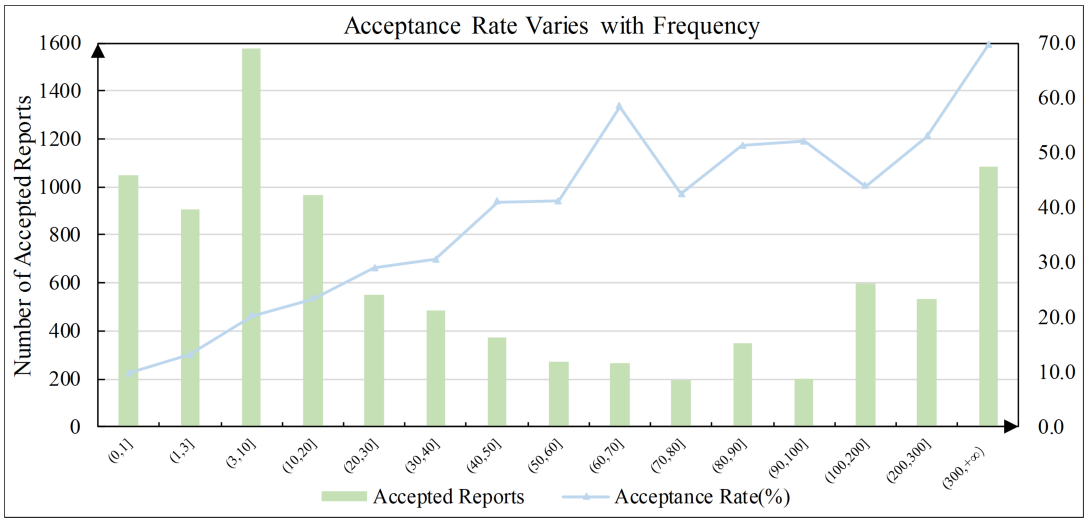

Figure 6 Acceptance Rate with Frequency

cross-application developer and inner-application developer were 24.72%, 55.61%, respectively, while the acceptance rate of regular users was 10.30%. This is consistent with the survey result by Heppler et al. based on the IBM data, that is, reports proposed by developers have higher possibility of being accepted by the manager[11]. In addition, we investigate the relationship between acceptance rate and frequency. We count the frequency for each creator, that is, the number of proposed reports in our data set. Then we group creators into different groups according to their frequency. The number of accepted reports by creators in each group and the proportion to the total is shown in Fig. 6. The figure shows that creators with higher frequency have higher acceptance rate. This can be explained from two folds. On the one hand, creators with higher frequency have higher possibility of being stakeholders of a application. On the other hand, creators with higher frequency may be more proficient in making reports that are easily accepted. Therefore, frequency is an effective indicator. To make full use of creator profile, we also use creator's ID as a label of the report. So that reports proposed by the same creator are linked by creator's ID. In this way, the model can learn the creator's information more exhaustively.

RQ4. Does re-sampling help to improve the performance of the proposed approach? If yes, to what extent?

Method. Our data set is unbalanced, which may cause bias to the classification results. In order to correct the bias, we employ re-sampling to adjust the category distribution. We adopt both under-sampling and over-sampling strategies on our data set. We use the *RandomOverSampler* and *RandomUnderSampler* in the Imblearn library [8]. The testing data remains as it is, and by copying the minority data or deleting the majority data, we balance the distribution of the two categories. The combination of CNN and BERT can achieve the best result. Therefore, we only experiment the re-sampling strategies with CNN classifier and BERT language model.

Results. The re-sampling results of cross-application and non-cross-application are shown in Table 7. After re-sampling, the accuracy both declines. The accuracy for cross-application is 77.8% and 79.1%, respectively for under-sampling and over-sampling, which are 2.9% and 1.6% lower. For non-cross-application is 78.5% and 82.4%, reduce by 5.1% and 1.2%, respectively. For approved reports, the precision declines with re-sampling strategies. But the recall and F measure are both improved. For rejected reports, it is on the contrary. The precision improves and the recall and F measure decline with re-sampling strategies. When comparing under-sampling and over-sampling strategies, over-sampling strategy can have high accuracy than under-sampling strategy. Besides, over-sampling strategy has higher precision and lower recall and F measure for approved reports. But it is on the contrary for rejected reports.

[8]https://imbalanced-learn.org/stable/

Table 7 Re-sampling Results of Both Cross-application and Non-cross-application.

		Acc	Approved(%)			Rejected(%)		
			P	R	F	P	R	F
Cross	No	80.7	67.6	34.2	43.6	82.1	93.4	87.3
	Under	77.8	51.9	59.8	54.2	87.2	80.4	83
	Over	79.1	57.8	42.9	48	83	89.4	85.9
Non-cross	No	83.6	70.5	50.5	58.8	86.3	93.6	89.8
	Under	78.5	52.5	73.4	61.2	90.9	79.9	85.1
	Over	82.4	62.8	59.5	61.1	88	89.4	88.7

Table 8 Results for each fold.

Fold	Non-cross-application(%)							Testing Application	Cross-application(%)						
		Approved			Rejected					Approved			Rejected		
	Acc	P	R	F1	P	R	F1		Acc	P	R	F1	P	R	F1
1	83.3	69	51.4	58.9	86.3	93	89.6	Bugzilla	71	72.9	**61.6**	**66.8**	69.7	79.4	74.2
2	83.4	72	46.7	56.6	85.4	**94.5**	89.7	SeaMonkey	89.3	58.8	26.8	36.8	91.1	97.6	94.2
3	82.6	67.3	49.3	56.9	85.8	92.8	89.2	Core Graveyard	78.3	55.1	39.8	46.2	83.1	90.1	86.4
4	84.1	70.9	**53.5**	**61**	**86.9**	93.4	90	Core	72.4	67.4	54.9	60.5	74.7	83.4	78.8
5	83.9	71.5	51.5	59.9	86.5	93.8	90	MailNews Core	82.7	73.8	19.4	30.7	84.2	98.4	90.7
6	83.6	70.8	50.2	58.7	86.2	93.7	89.8	Toolkit	82.6	66.1	40.3	50.1	85.2	94.3	89.5
7	83.6	70.4	50.6	58.9	86.3	93.6	89.8	Firefox	90	64.2	30.4	41.3	91.6	97.8	94.6
8	83.4	68.8	51.3	58.8	86.4	93	89.6	Thunderbird	**91.7**	**75.8**	30.6	43.6	**92.4**	**98.9**	**95.5**
9	**84.3**	**72.4**	51.2	60	86.5	94.1	**90.2**	Calendar	75.1	73	21.7	33.4	75.3	96.8	84.7
10	83.8	71.7	49.3	58.4	86.1	94.2	89.9	Camino Graveyard	73.6	69	16.7	26.9	73.8	96.9	83.8
Average	83.6	70.5	50.5	58.8	86.3	93.6	89.8	Average	80.7	67.6	34.2	43.6	82.1	93.4	87.3

Discussion. The experimental results indicate that re-sampling can help to distinguish minority category more precisely, but it does not work for the majority category. Although re-sampling does help to reduce the bias of categories, but more reports are predicted as rejected and the recall of rejected has declined. According to our experimental results, re-sampling strategies are not effective in improving the accuracy in this scenario. So in future studies, we do not recommend re-sampling strategies. Or at least, re-sampling strategy should be chosen according to the classification purpose.

RQ5. What's the performance of our proposed approach on each fold?

Method. As the experimental results show in RQ3, CNN and BERT achieved the best performance among all the classifiers in our context. In order to explore the performance of the model in more detail, we list the detailed results of the classifier on each fold.

Results. The detailed results for each fold are listed in Table 8. For non-cross-application evaluation, the accuracy results distribute between 82.6% and 84.3%. The precision, recall and F1 measure for accepted reports are between 67.3% and 72.4, 46.7% and 53.5%, 56.6% and 61%, while that for rejected reports are between 85.4% and 86.9%, 92.8% and 94.5%, 89.2% and 90.2%, respectively. The accuracy of the fourth and ninth fold is above 84%. Only the third fold is below 83%. For cross-application, the accuracy is between 71% and 91.7%. The precision, recall and F1 measure for accepted reports are between 55.1% and 75.8%, 16.7% and 61.6%, 26.9% and 66.8%, and that for rejected reports are between 69.7% and 92.4%, 79.4% and 98.9%, 74.2% and 95.5%. The prediction of Thunderbird achieves the best results, which is 91.7%. The prediction of Bugzilla achieves the lowest overall accuracy, which is 71%. But the F1 score for recall is higher than all other applications. Overall, the average results of non-cross-application are better than cross-application, almost 3% higher in accuracy.

Discussion. For non-cross-applications, the results are fairly evenly distributed in terms of evaluation metrics. The reason is that the training data and testing data are split according to application. The text features in the training data and testing data are evenly distributed. So there is not significant difference between different folds. So we can say that our model is stable under a consistent corpus. However, for cross-application, the evaluation results for different folds (applications) vary. The reason is that different applications possess different semantic and features, thus the cross-prediction results between applications may vary. Compared to cross-application prediction, non-cross-application incorporate data from

the same application into training set, thus the model can learn more from such data. The comparison between cross-application and non-cross-application indicates that to obtain better results, we should incorporate as much data from the same application in the training set as possible, to ensure that the model can learn from a similar corpus, as well as a stable performance.

5 | THREATS TO VALIDITY

5.1 | External Validity

Although we mitigated the external threats to validity by using a sample with sufficient number of enhancement reports (with 40,551 enhancement reports from 10 open sourced applications), which has also been adopted by a series of previous studies [2,4,6,7,5,8]. However, the data set only consists of open-source projects. Thus, it would be beneficial to also perform similar studies on industrial projects. Moreover, the approach is not a one-fit all solution, but only suits with projects that involve developers in filing enhancement reports. When it comes to the external validity brought by the model, we employ 10-fold cross-validation strategy and report the experimental results on both cross-application and non-cross-application.

5.2 | Internal Validity

In terms of internal validity, the reproduction of existing approaches may affect the experimental results. Because their approaches are not open source. To mitigate the threat, we have two co-authors double checked the implementation of the existing approaches and choose the most suitable parameters in our experiments. However, there may still be some differences in terms of parameters.

5.3 | Construct Validity

The pivotal part of our model is the calculation of creator profile. In order to make the creator profile more reliable, we only take advantage of the most fundamental information, which can be easily calculated from the metadata of enhancement reports, including the *creator* and *assigned_to*. The role of the creator is usually not directly tagged in the ITS. We assume that a person who has previously been assigned to an enhancement report to be a developer. In this mode, we can ensure the correctness of developer set but may miss some. In actual scenarios, the managers must know well about the role of the creator.

6 | CONCLUSION AND FUTURE WORK

In this paper, we propose a creator profile based method for ERAP task, which is based on TextCNN classifier and BERT embedding. We investigate the performance of different deep learning models and language models on ERAP task. We employ BERT to embed the enhancement reports and TextCNN to extract semantic information. Besides, we also use creator profile to improve the performance. We are the first to put forward the concept of creator profile, which includes: ID, Role and Frequency of the creator. Experimental results prove that creator profile is an effective feature for ERAP, which can improve the accuracy of existing methods by 4%. Compared to the stats-of-the-art approaches, our method gained higher accuracy, precision, recall and F-measure. Most importantly, creator profile is able to improve all the existing methods in terms of the evaluation metrics. In this way, we successfully identified and validated a novel and effective feature for predicting valuable user feature requests, which holds promise for application in other similar tasks as well. In practical scenarios, project managers are aware of which individuals are developers, Based on this knowledge, they can construct more precise creator profiles, ensuring their effectiveness and usefulness.

Our first attempt on creator profile opens up new way for future research. In future, we would like to involve more user characteristics for a more vivid creator profile, e.g., more historical activity of creators. Also, we would like to collect more data from different ITSs to evaluate our approach. Furthermore, we would also like to exploit more text features to improve the performance of the approach. There is still great improvement space for ERAP task.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (No. 61802167), and the Fundamental Research Funds for the Central Universities.

References

1. Hastie S, Wojewoda S. Standish group 2015 chaos report.; 2015.
2. Nizamani ZA, Liu H, Chen DM, Niu Z. Automatic approval prediction for software enhancement requests. *Autom. Softw. Eng.* 2018; 25(2): 347–381. doi: 10.1007/s10515-017-0229-y
3. Erne M, Jiang Z, Liu V. Investigating the Effect of User Reviews on Mobile Apps: The Role of Customer Led Innovation. In: Springer. ; 2020: 193–200.
4. Umer Q, Liu H, Sultan Y. Sentiment based approval prediction for enhancement reports. *J. Syst. Softw.* 2019; 155: 57–69. doi: 10.1016/j.jss.2019.05.026
5. Nafees SA, Ur Rehman FA. Machine learning based approval prediction for enhancement reports. In: ; 2021: 377-382
6. Arshad MA, Zhiqiu H. Using CNN to Predict the Resolution Status of Bug Reports. In: . 1828. IOP Publishing. ; 2021: 012106.
7. Arshad MA, Huang Z, Riaz A, Hussain Y. Deep Learning-Based Resolution Prediction of Software Enhancement Reports. In: IEEE. ; 2021: 0492–0499.
8. Cheng J, Sadiq M, Kalugina OA, Nafees SA, Umer Q. Convolutional Neural Network Based Approval Prediction of Enhancement Reports. *IEEE Access* 2021; 9: 122412–122424.
9. Li C, Huang L, Ge J, Luo B, Ng V. Automatically classifying user requests in crowdsourcing requirements engineering. *J. Syst. Softw.* 2018; 138: 108–123. doi: 10.1016/j.jss.2017.12.028
10. Niu F, Li C, Ge J, Luo B. A Survey of User Feature Requests Analysis and Processing. *Journal of Software* 2022; 33: 1–32.
11. Heppler L, Eckert R, Stuermer M. Who cares about my feature request?. In: Springer. ; 2016: 85–96.
12. Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* 2018.
13. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* 2013.
14. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural computation* 1997; 9(8): 1735–1780.
15. Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. In: ; 2014: 1532–1543.
16. Shi L, Chen C, Wang Q, Li S, Boehm B. Understanding feature requests by leveraging fuzzy method and linguistic analysis. In: IEEE. ; 2017: 440–450.
17. Jayatilleke S, Lai R, Reed K. A method of requirements change analysis. *Requir. Eng.* 2018; 23(4): 493–508. doi: 10.1007/s00766-017-0277-7
18. Morales-Ramirez I, Munante D, Kifetew F, Perini A, Susi A, Siena A. Exploiting user feedback in tool-supported multi-criteria requirements prioritization. In: IEEE. ; 2017: 424–429.
19. Zhang T, Chen J, Zhan X, Luo X, Lo D, Jiang H. Where2change: Change request localization for app reviews. *IEEE Transactions on Software Engineering* 2019; 47(11): 2590–2616.
20. Xu C, Sun X, Li B, Lu X, Guo H. MULAPI: Improving API method recommendation with API usage location. *Journal of Systems and Software* 2018; 142: 195–205.
21. Calefato F, Lanubile F, Maiorano F, Novielli N. Sentiment Polarity Detection for Software Development. *Empirical Software Engineering* 2018; 23(3): 1352–1382.
22. Niu F, Li C, Chen H, Ge J, Luo B. Towards Just-In-Time Feature Request Approval Prediction. In: ; 2022: 262–271.
23. Navarro-Almanza R, Juarez-Ramirez R, Licea G. Towards supporting software engineering using deep learning: A case of software requirements classification. In: IEEE. ; 2017: 116–120.

24. Shi L, Xing M, Li M, Wang Y, Li S, Wang Q. Detection of hidden feature requests from massive chat messages via deep siamese network. In: ; 2020: 641–653.

25. Liu H, Xu Z, Zou Y. Deep learning based feature envy detection. In: ; 2018: 385–396.

26. Malik MSI. Predicting users' review helpfulness: the role of significant review and reviewer characteristics. *Soft Comput.* 2020; 24(18): 13913–13928. doi: 10.1007/s00500-020-04767-1

27. Gill KD, Raza A, Zaidi AM, Kiani MM. Semi-automation for ambiguity resolution in Open Source Software requirements. In: IEEE. ; 2014: 1–6.

28. Loper E, Bird S. Nltk: The natural language toolkit. *arXiv preprint cs/0205028* 2002.

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

30. Janák J. Issue tracking systems. *Brno, spring* 2009: 17.

AUTHOR BIOGRAPHY

Feifei Niu. Feifei Niu is a PhD candidate at the Software Institute, Nanjing University, under the supervision of Assistant Professor Chuanyi Li and Prof. Bin Luo. Her research interests include software engineering, requirements engineering, defect prediction, fault localization, and process mining. She can be contacted at niufeifei0624@gmail.com.

Chuanyi Li. Chuanyi Li is an Assistant Researcher at Software Institute, Nanjing University. He received the BS and PhD degrees from Nanjing University in 2012 and 2017, respectively. His research interests include software engineering, systems and software quality assurance, process modeling, simulation and improvement, process mining.

Heng Chen. Heng Chen is a master student at Software Institute, Nanjing University. His research interests include natural language processing and software engineering.

Jidong Ge. Jidong Ge is an Associate Professor at Software Institute, Nanjing University. He received his PhD degree in Computer Science from Nanjing University in 2007. His current research interests include workflow modeling, process mining, cloud computing, workflow scheduling, software engineering. His research results have been published in more than 90papers in international journals and conference proceedings including IEEE TPDS, IEEE TSC, JASE, COMNET, JPDC, FGCS, JSS, Inf. Sci., JNCA, JSEP, ESA, ICSE, IWQoS etc

Bin Luo. Bin Luo is a full Professor at the Software Institute, Nanjing University. His main research interests include cloud computing, computer network, workflow scheduling, and software engineering. His research results have been published in more than 50 papers in international journals and conference proceedings including IEEE TSC, ACM TIST, JSS, FGCS, Inf Sci, ESA, ICTAI, etc. He is leading the institute of applied software engineering at Nanjing University.

Alexander Egyed. Prof. Dr. Alexander Egyed is a Full Professor at the Johannes Kepler University (JKU), Austria and the Chair for Software-Intensive Systems. Before joining the JKU, he worked as a Research Scientist for Teknowledge Corporation, USA (2000-2007) and then as a Research Fellow at the University College London, UK (2007-2008). He received a Doctorate degree from the University of Southern California, USA. His research interests are diverse and broadly center around software system design and modeling, variability management, requirements engineering, consistency checking and resolution, traceability, and change impact analysis. He is a member of ACM, ACM SigSoft, IEEE, and IEEE Computer Society.

How to cite this article: Feifei Niu, Chuanyi Li, Heng Chen, Jidong Ge, Bin Luo and Alexander Egyed (2023), Utilizing Creator Profiles for Predicting Valuable User Enhancement Reports, *Software: Evolution and Process*, *2023;00:1–20*.