IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

ADate of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Convolutional Neural Network Based Classification of App Reviews

NAILA ASLAM[1], WAHEED YOUSUF RAMAY[2], XIA KEWEN[1], AND NADEEM SARWAR[3]
[1]School of Electronics and Information Engineering, Hebei University of Technology, Tianjin 300401, China. (e-mail: nailaaslam2020@yahoo.com, kwxia@hebut.edu.cn)
[2]COMSATS University Islamabad, Pakistan. (e-mail: waheedramaycs@gmail.com)
[3]Bahria University, Lahore, Pakistan. (e-mail: nadeem_srwr@yahoo.com)

Corresponding author: Xia Kewen (e-mail: kwxia@hebut.edu.cn).

ABSTRACT An app store (i.e., *Google Play*) is a platform for mobile apps for almost every software and service. App stores allow users to browse and download apps and facilitate developers to keep an eye on their apps by providing ratings and reviews of the apps. App reviews may include the user's experience, information about bugs, request for new features, or rating of the app in word. The manual categorization of app reviews is critical and time-consuming for developers. Automatic classification of app reviews may help developers especially for fixing bugs on time. In this perspective, several approaches have been proposed for the automatic classification of reviews. However, none of them exploits the non-textual information of app reviews. In this paper, we propose a deep learning based approach for the classification of app reviews. It does not only leverage non-textual information of app reviews but also exploits a deep learning technique that has proved more accurate for the text classification in various domains. The approach first extracts textual and non-textual information of each app review, preprocesses the textual information, computes the sentiment of app reviews using Senti4SD, and determines the history of the reviewer includes the total number of reviews posted by the reviewer, and his submission rate (i.e., what percentages of his review have been submitted for the associated app). Second, we create a digital vector against each app review. Finally, we train a deep learning based multi-class classifier to classify app reviews. The proposed approach is evaluated on a public dataset, and the results suggest that it significantly improves the state of the art. It improves average precision from 75.72% to 95.49%, average recall from 69.40% to 93.94%, and f-measure from 72.41% to 94.71%, respectively.

INDEX TERMS Classification, Convolutional Neural Network, Machine Learning, Sentiment, User Feedback, Mobile App Reviews.

I. INTRODUCTION

IN this digital world, softwares are moved from computers to mobile phones. App stores (i.e., *Google* Play Store and *Apple* AppStore) provide mobile apps (noted as *apps* for short in the rest of this paper) almost for every field of life. In the fourth quarter of 2019, Google's Play Store and Apple's AppStore were the top two largest app stores with 2.57 million and 1.84 million apps, respectively[1]. Such app stores enable users to browse and download apps, and collect users' reviews (i.e., star rating and textual feedback) for developers to improve their apps.

The existing study [1] also indicates the importance of app reviews (noted as *reviews* for short in the rest of the paper) for the success of any mobile app. The ranking of apps may increase with better reviews, better visibility, higher sales, and download numbers [2]. In addition, recent researches indicate the importance of reviews for developers. They notice that reviews may include users' experience [3], information about bugs [4], request for new features [5],

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3029634, IEEE Access

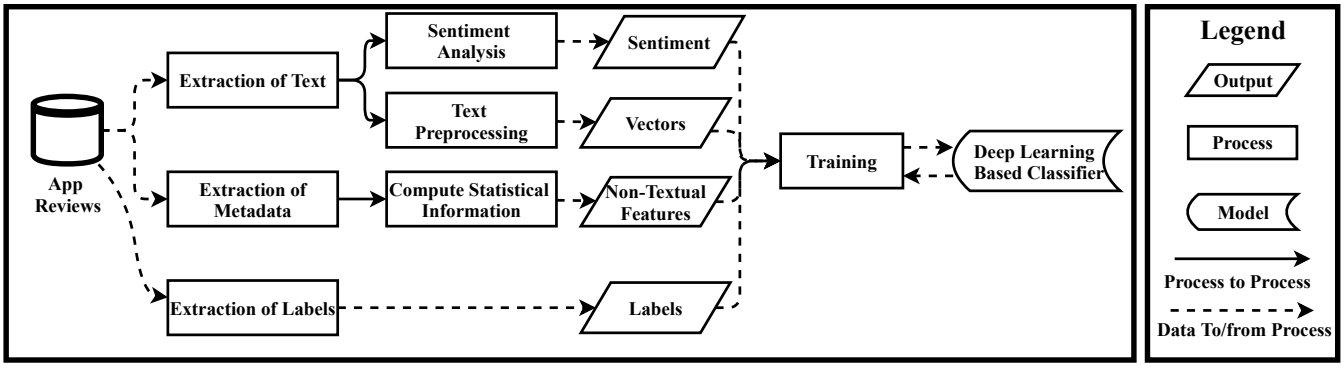

FIGURE 1. Overview of the Proposed Approach

or textual information for rating [3], [6]. Moreover, many reviews are low-quality that include meaningless information, spam, or star rating in words. Developers manually categorize such reviews that is critical and time-consuming. Automatic classification of reviews helps developers for adding new features to increase the popularity of their apps and/or resolving bugs on time for the maintenance of their apps [7].

To this end, a number of automated approaches have been proposed for automatic classification of reviews [7]–[10]. Such approaches consider app reviews as plain texts, their metadata (e.g., length of text) and sentiment, and employ traditional machine learning techniques to make the prediction. To further improve their performance, in this paper, we propose a Convolutional Neural Network (CNN) based approach for the classification of app reviews. On one hand, it leverages non-textual information of app reviews that have not yet been employed by existing approaches. On the other hand, it exploits a deep learning based classifier that has proved more accurate for the text classification in various domains.

The approach works as follows: 1) it extracts textual (i.e., the textual information, and the sentiment of the textual information computed by *Senti4SD*) and non-textual features (i.e., the statistics of the reviewer (the total number of reviews posted by the reviewer, and his submission rate, i.e., what percentages of his review have been submitted for the associated app), and the statistics (metadata) of each app review; 2), it preprocesses the textual information and transforms it into a digital vector; and 3) it trains a CNN classifier to classify multi-class reviews. The proposed approach is evaluated on a public dataset, and the results suggest that it significantly improves the state of the art. It improves average precision from $75,72\%$ to 95.49%, average recall from 69.40% to 93.94%, and f-measure from 72.41% to 94.71%, respectively.

The rest of the paper is organized as follows: Section II provides the details of the proposed approach. Section III presents the evaluation of the proposed approach. Section IV introduces related work, and Section V concludes the paper and indicates future interests.

II. APPROACH

A. OVERVIEW

Identification of the associated class of app reviews (noted as $reviews$) is essentially a multi-class classification. All the submitted reviews are automatically classified into four classes, i.e., bug reports, enhancement reports, user experiences, and ratings.

Fig. 1 illustrates an overview of the proposed approach. A brief introduction is presented as follows:

1) For each review r, we extract its textual information (noted as t_r), i.e., the text of the review.
2) We extract non-textual features (noted as nt_r) of the review, e.g., the statistical data of the app, i.e., app size, the number of app installations, and the statistical data of the reviewer.
3) The textual information t_r is preprocessed with natural language processing techniques, and convert it into numerical vectors.
4) We calculate the sentiment (noted as s_r) of each review. We compute the sentiment based on the textual information tr of the review.
5) We extract the labels of the reviews, i.e., whether they belong to bug reports, feature requests, user experiences, or ratings.
6) We train a multi-class classifier with the labeled data that are collected in the previous steps.
7) Finally, for new reviews, we extract their t_r and nt_r (as mentioned in Steps 1-4), and input to the trained multi-class classifier to generate the label (bug reports, enhancement reports, user experiences, or ratings) of the new reviews.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3029634, IEEE Access

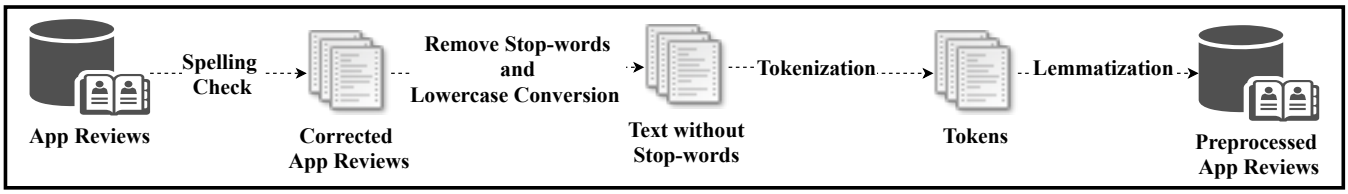

FIGURE 2. Overview of the Preprocessing

Details of the proposed approach are presented in the following sections.

B. DATA EXTRACTION

A review r from a set of reviews R can be defined as follows:

$$r =< t_r, nt_r, l > \quad (1)$$

where tr is the textual information (i.e., text) of the review, nt_r is the non-textual information of the review, and l is the category (label) of r, i.e., whether it belongs to *bug reports*, *enhancement report*, *user-experience*, or *rating*.

Non-textual information holds statistics of the reviewer who committed the review and statistics of the app associated with the review:

$$nt_r = < SRev, SApp > \quad (2)$$

$$SRev = < Rev_n, Rev_r > \quad (3)$$

$$SApp = < App_s, App_i > \quad (4)$$

nt_r consists of two metrics ($SRev$ and $SApp$), where $SRev$ further consists of two metrics: Rev_n and Rev_r. Rev_n is the number of reviews (either *bug reports*, *enhancement reports*, *user experience*, or *rating*) associated with the given app, and Rev_r is the average rate for review (correspond to same review category) associated with the app.

$SApp$ also consists of two metrics (App_s and App_i) where App_s is the size of the app and App_i is the number of installation of the app. $SApp$ is collected from the metadata of the app.

Labels (*bug report*, *enhancement report*, *user experience*, and *rating*) are defined using the most trivial technique that checks the particular list of words within each review to automatically classify it. In this regard, we use SQL queries, i.e., LIKE. Table 1 shows a list of words (the complete list can not be represented, therefore we only represent the most influential word) for each category of review. These lists are formed based on the existing researches [3], [5], [11]. The existing researches consider *nouns* (aspects), *adjectives*, *verbs*, and *adverbs* as keywords. Note that, one review may belong to one or more categories, e.g., "Doesn't work at the moment. Was quite satisfied before the last update. Will change the rating once it's functional again" can be categorized as a *bug report* or *rating review*.

TABLE 1. Top Keywords for Categorization of Reviews

Review Category	Keywords
Bug Reports	fix, bug, issue, crash, defect
Enhancement Reports	request, suggest, add, want, wish
User-experience Reviews	support, assist, situation, help, improve
Rating Reviews	good, love, bad, very, worst

C. PREPROCESSING

We preprocess the textual information t_r to convert it into a digital vector. Fig. 3 illustrates an overview of the preprocessing. The details of the preprocessing are as follows.

First, we apply the spelling check on each review. Then, we extract and remove the *stop-words*, e.g., special characters from reviews and convert the remaining text into lowercase to improve the comparison. After that, we tokenize each review into tokens (words) using *Python NLTK* [12]. Next, we *lemmatize* the tokens to turn the comparative and superlative words into their base words, e.g., *liked* turns into *like*). Finally, we convert the preprocessed text into vectors (embedding). In this regard, we leverage *word2vec* [13] that takes each token from the preprocessed text and converts it into a fixed-length numeric vector. Notably, we exploit standard libraries *Gensim* and *TensorFlow* to implement *word2vec*. We implement the *Skip-gram* architecture of *word2vec* for the given dataset with settings: *window_size = 2*, *n=300* (dimensions of word embeddings), *epoch=50*, and *learning_rate=0.01*. A concatenated vector WV_r is created for each from the vectors of each token of the review.

D. SENTIMENT ANALYSIS

The existing studies [14], [15] and the word-list of each category (as shown in Table 1) motivate and suggest that sentiment related words may help in the classification of reviews. For example, 79.6% of the rating reviews contains the keywords "*good*" and "*bad*".

We compute the sentiment $RSen$ of each review and consider it as one of the key feature of the reviews. It can be represented as:

$$RSen(r) = CalRSen(r.t_r) \quad (5)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3029634, IEEE Access

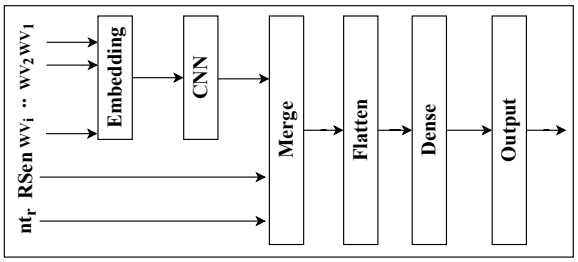

FIGURE 3. Overview of the Classifier

where r is a review, $r.t_r$ is the textual feature of the review, and $RSen(r)$ is the sentiment of the review.

The function $CalRSen$ computes the sentiment of each review r using *Senti4SD* [16]. It exploits three different kinds of features: 1) sentiment lexicon; 2) n-grams extracted from the given dataset (uni-gram and bi-gram in our case); and 3) semantic features. Semantic features are dependent on word representation in a distributional semantic model. The semantic features capture the similarity between the vector representations of the Stack Overflow documents and prototype vectors representing the polarity classes in a distributional semantic model, built using positive, negative, and neutral words from the sentiment lexicon. *Senti4SD* considers the emotion-words, modifiers, and negation in the review for calculation and returns the sentiment of the review. Note that, we leverage *Senti4SD* because of its significant performance for software engineering text in contrast to most commonly used repositories, e.g., *SentiWordNet* [17].

E. CONVOLUTIONAL NEURAL NETWORK BASED CLASSIFIER

The proposed Convolutional Neural Network (CNN) based classifier first extracts the features from preprocessed textual information WV_r and combines the extracted textual features and non-textual features nt_r, and predicts labels based on the combined features. The details of both steps are as follows.

1) Extraction of Textual Features

The CNN based classifier takes WV_r and returns a feature map of c that contains the maximum values of the features. The classifier exploits the three convolution layers for the extraction of textual features. We apply filter sizes 3, 4, and 5, respectively. Each filter improves feature vectors by performing a convolution on the corresponding layer and create a feature map for the next layer.

The initial convolution layer takes a word WV_i from WV_r of a k-dimensional vector, where k is equal to 300. Assume $x_i \epsilon X^k$ be the k-dimensional vector corresponding to WV_i and $X_{i:i+j}$ is the concatenation of feature vectors x_i - X_{i+j}. A filter $\mathbf{w} \epsilon X^{dk}$ is applied to a window of d words creates a new feature map c_i that can be defined as,

$$c_i = f(\mathbf{w} \cdot WV_{i,i+d-1}) \tag{6}$$

where b represents a bias and f represents the tangent non-linear function. The filter creates a feature map $\mathbf{c}$ using the given window of features that can be defined as

$$c = < c_1, c_2, \ldots, c_i > \tag{7}$$

Notably, we pass the numerical vectors into a CNN with $dropout = 0.2$ to prevent the overfitting.

2) Classification Based on Merged Features

The classifier takes the additional inputs: the sentiment of each review $RSen$ and the statistical information of the app $SApp$, and merge the additional inputs with the processed textual information using the merge layer. The merge layer directly fuses $RSen$ and $SApp$ in the processed textual information $\mathbf{c}$. Note that we merge the additional information directly to reduce the loss of convolutions in contrast to passing it into a separate network. Then, the flatten layer converts the integrated feature map matrix into a vector. Finally, dense layer takes the flattened information, computes the weighted average $\mathbf{w}$ and a bias b of the integrated features, and applies a non-linear activation function $relu$ to predicts the classes.

III. EVALUATION

In this section, the proposed deep learning-based classification approach (*DCAR*) for reviews is evaluated with the real-world reviews from *Google Play* and *Apple* app stores.

A. RESEARCH QUESTIONS

We investigate the following research questions for the evaluation of *DCAR*.

- **RQ1:** Can *DCAR* outperform the state of the art in the classification of reviews? If yes, to what extent?
- **RQ2:** How do different features of reviews influence the performance of *DCAR*?
- **RQ3:** How does the preprocessing influence the performance of *DCAR*?
- **RQ4:** Does the proposed classifier outperform the other classifiers in the classification of reviews?

B. DATASET

We exploit the reviews dataset which is extracted by Maalej et al. [7] from *Google* store and *Apple* store. The review only from the top apps are crawled. The total 1,126,453 reviews for 1100 apps from *Apple*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3029634, IEEE Access

TABLE 2. Dataset Statistics

	Application Name	Category	No. of Reviews	No. of Samples
Apple Store	**1100 apps**	iOS	1,126,453	1000
	Dropbox	Productivity	2009	400
	Evernote	Productivity	8878	400
	TripAdvisor	Travel	3165	400
Google Store	**80 apps**	android	146,057	1000
	PicsArt	Photography	4438	400
	Pinterest	Social	4486	400
	Whatsapp	Communication	7696	400

store, and 146,057 reviews for 80 apps from *Google* store. Each review contains *text*, *title*, *app name*, *category*, *store*, *date of submission*, *reviewer-id*, and *rating*.

We follow Maalej et al. [7] and consider their manually labeled dataset of 4400 reviews for the evaluation of *DCAR*. The data is selected in two phases. In the first phase, 2000 reviews are randomly selected from the collected reviews from both stores (1000 reviews from each store). In the second phase, the top 3 apps are first selected from both stores. Then, 400 reviews are randomly selected from each app. In total, four sets of sample reviews (named as 1100 apps, Dropbox, Evernote, and TripAdvisor) are created from the *Apple* store reviews, where the sets contain 100, 400, 400, and 400 reviews, respectively. Similarly, four sets of sample reviews (named as 80 apps, PicsArt, Pinterest, and Whatsapp) are created from the *Google* store reviews, where the sets contain 100, 400, 400, and 400 reviews, respectively. The statistics of the dataset are shown in Table 2.

Moreover, they conducted a paid *peer, manual content analysis* for the selected reviews to create the truth set. They sent every review to 2 randomly selected coders out of 10 computer science experts. They briefly explained the coders in a meeting about the manual classification task and provided a tool for classification. The manually analyzed reviews contain 2000 randomly selected app (1000 Apple apps and 1000 Google apps) and 2400 manually selected apps (1200 Apple apps and 1200 Google apps).

C. EXPERIMENTAL DESIGN

1) RQ1:Comparison against the State of the art

The first research question (RQ1) provides a comparison between the proposed approach (*DCAR*) and the state of the art. To answer RQ1, we compare *DCAR* against the Maalej approach [7] (noted as Maalej's for short in the rest of this paper). To the best of our knowledge, Maalej's reports the best results for the classification of reviews. We also compare *DCAR* against Umer approach [15] and Ramay approach [19] as both are declared best for the classification software engineering text.

The comparison employs the ten-fold cross-validation for the evaluation. On each fold, reviews from a single set of sample reviews (10%) are taken for testing (noted as $sTest$), whereas others (90%) are taken for training (noted as $sTrain$). We train *DCAR* and *Maalej's* separately with the same $sTrain$. After that, the trained models are evaluated separately with the same $sTest$. Note that, we evaluate the performance of *DCAR* and *Maalej's* using the well-known and most adopted metrics for machine learning classification [7], [14], [15], [18], i.e., *precision*, *recall*, and *f-measure*.

2) RQ2: Influence of Different Features

The second research question (RQ2) examines the influence of different features employed by *DCAR* as mentioned in Section II-B. *DCAR* leverages t_r and nt_r features of reviews. We disable each of them and repeat the evaluation (only for *DCAR*) as mentioned in Section III-C1. Such evaluation measures the impact of each features on *DCAR*.

3) RQ3: Influence of Preprocessing

The third research question (RQ3) examines the influence of the preprocessing (Section II-C) on the given dataset by comparing the performance of *DCAR* with preprocessing disabled *DCAR*. We remove all the preprocessing steps and repeat the evaluation (as mentioned in Section III-C1) to examine the impact of preprocessing.

4) RQ4: Comparison among Different Classifiers

The fourth research question (RQ4) provides a comparison of the proposed classifier among other machine and deep learning classifiers. In this regard, we exploit the Naive Bayes (NB), Multi-nomial Naive Bayes (MNB), Decision Tree (DT), Support Vector Machine (SVM), Convolutional Neural Network (CNN), and Long Short Term Memory (LSTM) and repeat the evaluation as mentioned in Section III-C1. Note that we select these classifiers due to their significant performance for the textual classification [7], [14], [15], [18], [19].

D. RESULTS

1) RQ1:Comparison against the State of the art

To answer the research question RQ1, we compare *DCAR* against *Maalej's*, *Umer's*, and *Ramay's*. Table 3 presents the evaluation results. The first column represents the approaches, the second column represents the evaluation metrics, and columns 3-6 present the performance on each of the given categories. The last column presents the average performance of the approaches of each testing category.

TABLE 3. Performance of the Proposed Approach

		Bug Reports	Enhancement Reports	User Experiences	Ratings	Average
Proposed Approach	**Precision**	**94.69%**	**95.71%**	**95.76%**	95.78%	**95.49%**
	Recall	**94.02%**	**93.89%**	**93.94%**	**93.91%**	**93.94%**
	F-measure	**94.35%**	**94.79%**	**94.84%**	**94.84%**	**94.71%**
Maalej's Approach	**Precision**	83.16%	82.59%	84.03%	**96.30%**	86.52%
	Recall	84.65%	79.71%	85.36%	87.26%	84.25%
	F-measure	83.90%	81.12%	84.69%	91.56%	85.32%
Umer's Approach	**Precision**	76.30%	79.20%	75.27%	72.12%	75.72%
	Recall	69.54%	70.06%	69.23%	68.77%	69.40%
	F-measure	72.76%	74.35%	72.12%	70.41%	72.41%
Ramay's Approach	**Precision**	81.42%	81.98%	81.31%	79.96%	81.17%
	Recall	79.56%	80.06%	80.88%	80.05%	80.14%
	F-measure	80.48%	81.01%	81.09%	80.00%	80.65%

The first row represents the testing category, and the rest of the rows present the performance of the approaches on the given category. The table presents the best performance for each testing category in bold.

From Table 3, the following observations are made.

- First, *DCAR* significantly improves the state of the art. Compared to *Maalej's*, *Umer's*, and *Ramay's*, the improvement of *DCAR* in average precision, average recall, and average f-measure is (**10.36%** = (95.49% - 86.52%) / 86.52%, **11.51%** = (93.94% - 84.25%) / 84.25%, and **11.00%** = (94.71% - 85.32%) / 85.32%), (**26.10%** = (95.49% - 75.72%) / 75.71%, **35.26%** = (93.94% - 69.40%) / 69.40%, and **30.79%** = (94.71% - 72.41%) / 72.41%), (**17.64%** = (95.49% - 81.17%) / 81.17%, **17.22%** = (93.94% - 80.14%) / 80.14%, and **17.43%** = (94.71% - 80.65%) / 80.65%), respectively.
- Second, concerning the f-measure, *DCAR* significantly outperforms *Maalej's*, *Umer's*, and *Ramay's* on every testing category. The improvement in f-measure varies from **3.58%** = (94.84% - 91.56%) / 91.56% to **16.85%** = (94.79% - 81.12%) / 81.12%, **27.49%** = (94.79% - 74.35%) / 74.35% to **34.70%** = (94.84% - 70.41%) / 70.41%, and **16.95%** = (94.84% - 81.09%) / 81.09% to **18.54%** = (94.74% - 80.00%) / 80.00%, respectively.
- Third, *DCAR* has significant improvement in recall on every testing category. The improvement in recall varies from **7.62%** = (93.91% - 87.26%) / 87.26% to **17.79%** = (93.89% - 79.71%) / 79.91%, **34.01%** = (93.89% - 70.06%) / 70.06% to **36.56%** = (93.91% - 68.77%) / 68.77%, and **16.15%** = (93.94% - 80.88%) / 80.88% to **18.17%** = (94.02% - 79.56%) / 79.56%, respectively. However, *DCAR* has slight reduction against *Maalej's* in precision on one testing category (*Ratings*), i.e., the reduction on *Ratings* is **0.54%** = (96.30% - 95.78%) / 95.78%. In the comparison of precision, *DCAR* has significant improvement against *Maalej's* in recall on the same testing category, i.e., the improvement on *Ratings* is **7.62%** = (93.91% - 87.26%) / 87.26%. Consequently, *DCAR* has significant improvement in f-measure on *Ratings*.

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
DCAR	4	3.788221273	0.947055318	5.54826E-06
Maalej's	4	3.412700091	0.853175023	0.001964198
Umer's	4	2.8964	0.7241	0.000265673
Ramay's	4	3.2258	0.80645	2.58167E-05

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.103917551	3	0.034639184	61.27477069	1.51155E-07	3.490294819
Within Groups	0.006783709	12	0.000565309			
Total	0.11070126	15				

FIGURE 4. ANOVA Analysis on F-measure

We perform one-way *ANOVA* on f-measure to further investigate the performance improvement of *DCAR*. *ANOVA* examines the difference between the performance of the given approaches. Fig. 4 illustrates the results of ANOVA analysis. The results suggests that *f-ratio* is 61.2748 and *p-value* is 1.5116E-07 that is less than 0.05. From Fig. 4, we conclude that *ANOVA* indicates a significant difference among the f-measure of the given approaches. Note that, we also conduct the *ANOVA* on precision and recall that confirms the significant improvement of *DCAR*.

Based on the preceding analysis, we conclude that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3029634, IEEE Access

DCAR significantly improves the state of the art in classification of reviews.

2) RQ2: Influence of Different Features

To answer the research question RQ2, we evaluate the performance reduction of *DCAR* by disabling a few of the given features. Table 4 presents the evaluation results. The first column presents the disabled features. The rest of the columns present the performance of *DCAR* against each disabled setting.

From Table 4, the following observations are made.

- First, the performance of *DCAR* reduces upon disabling any of the employed features. The default setting of *DCAR* (i.e., none of the features is disabled) achieves the highest average performance.
- Second, the textual features (i.e., extracted from reviews) is critical for *DCAR*. Disabling the textual features (2^{nd} row) returns the highest reduction in the performance of *DCAR*. The reduction in average precision, average recall, and average f-measure is 21.29% = 95.49% - 74.20%, 40.00% = 93.94% - 53.94%, and 32.24% = 94.71% - 62.47%, respectively.
- Third, non-textual features (i.e., statistics of reviewers and statistics of apps) are appropriate. Disabling non-textual features (5^{th} row) returns the performance reduction in average precision, average recall, and average f-measure by 4.48% = 95.49% - 91.01%, 3.45% = 93.94% - 90.49%, and 3.96% = 94.71% - 90.75%, respectively.
- Fourth, disabling statistics of apps (6^{th} row) has minor reduction in performance as compared to disabling statistics of reviewers (9^{th} row). The reduction of both cases in average precision, average recall, average f-measure is (0.12% = 95.49% - 95.37%, 0.11% = 93.94% - 93.83%, and 0.12% = 94.71% - 94.59%), and (4.09% = 95.49% - 91.40%, 3.00% = 93.94% - 90.94%, and 3.54% = 94.71% - 91.17%), respectively. The performance comparison of both cases indicates that statistics of reviews are more appropriate for the classification of reviews.
- Finally, we notice that disabling Rev_n (10^{th} row) has more significant reduction in performance in contrast to Rev_r (11^{th} row). The performance comparison of both cases indicates that Rev_n is more appropriate for the classification of reviews.

Based on the preceding analysis, we conclude that all of the employed features are useful. Leveraging the non-textual features, particularly the statistics of reviewers, results in a significant increase in performance.

TABLE 4. Influence of Different Features

	Disabled Features	Precision	Recall	F-measure
1#	**None**	**95.49%**	**93.94%**	**94.71%**
2#	**All Textual Features**	74.20%	53.94%	62.47%
3#	**Text**	70.49%	54.13%	61.24%
4#	**Sentiment**	92.48%	92.72%	92.60%
5#	**Non-textual Features**	91.01%	90.49%	90.75%
6#	**App's Statistics**	95.37%	93.83%	94.59%
7#	$\mathbf{App_s}$	95.44%	93.88%	94.65%
8#	$\mathbf{App_i}$	95.42%	93.86%	94.63%
9#	**Reviewer's Statistics**	91.40%	90.94%	91.17%
10#	$\mathbf{Rev_n}$	91.75%	91.30%	91.52%
11#	$\mathbf{Rev_n}$	92.23%	91.61%	91.92%

TABLE 5. Influence of Preprocessing

Preprocessing	Precision	Recall	F-measure
Enable	95.49%	93.94%	94.71%
Disable	94.40%	91.66%	93.01%

3) RQ3: Influence of Preprocessing

To answer the research question RQ3, we examine the influence of the preprocessing. We remove the preprocessing step and repeat the evaluation (as mentioned in Section III-C1). Table 5 presents the evaluation results. The second row presents the performance of *DCAR* with the default setting (i.e., preprocessing enabled). The third row presents the performance of *DCAR* without preprocessing (i.e., preprocessing disabled). The last row presents the performance improvement of *DCAR*.

From Table 5, the following observations are made.

- The proposed approach with the preprocessing achieves significant improvement in performance. The improvement in precision, recall, and f-measure is **1.15%** = (95.49% - 94.40%) / 94.40%, **2.49%** = (93.94% - 91.66%) / 91.66%, and **1.83%** = (94.71% - 93.01%) / 93.01%, respectively.
- Disabling the preprocessing step results in significant reduction in performance especially in recall by **2.43%** = (91.66% - 93.94%) / 93.94%.

From the preceding analysis, we conclude that the preprocessing is appropriate for the classification of reviews.

4) RQ4: Comparison against Different Classifiers

To answer the research question RQ4, we compare our neural network based classifier CNN with deep learning-based classifier (i.e., LSTM) and machine learning-based classifiers (i.e., Random Forest(RF),

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3029634, IEEE Access

TABLE 6. Performance of the Proposed Approach

		Bug Reports	Enhancement Reports	User Experiences	Ratings	Average
CNN	**Precision**	**94.69%**	**95.71%**	**95.76%**	**95.78%**	**95.49%**
	Recall	**94.02%**	**93.89%**	**93.94%**	**93.91%**	**93.94%**
	F-measure	**94.35%**	**94.79%**	**94.84%**	**94.84%**	**94.71%**
LSTM	**Precision**	90.85%	91.27%	92.08%	90.30%	91.13%
	Recall	89.02%	88.96%	88.92%	88.93%	88.96%
	F-measure	89.93%	90.10%	90.47%	89.61%	90.03%
SVM	**Precision**	88.56%	86.92%	87.11%	88.37%	87.74%
	Recall	80.64%	79.52%	79.99%	80.03%	80.05%
	F-measure	84.41%	83.06%	83.40%	83.99%	83.72%
MNB	**Precision**	77.82%	78.15%	77.91%	77.88%	77.94%
	Recall	83.11%	83.31%	84.60%	84.49%	83.88%
	F-measure	80.38%	80.65%	81.12%	81.05%	80.80%
NB	**Precision**	79.45%	81.01%	80.48%	79.96%	80.23%
	Recall	85.06%	84.97%	85.59%	85.32%	85.24%
	F-measure	82.16%	82.94%	82.96%	82.55%	82.65%
RF	**Precision**	89.00%	89.46%	89.52%	89.33%	89.33%
	Recall	90.05%	89.75%	89.39%	90.60%	89.95%
	F-measure	89.52%	89.60%	89.45%	89.96%	89.64%

Support Vector Machine (SVM), Multi-nomial Naive Bayes (MNB), and Naive Bayes (NB)). Table 6 presents the evaluation results. We bold the maximum performance of each classifier on each testing category.

From Table 6, the following observations are made.

- First, the proposed classifier (CNN) achieves the highest performance upon the selected deep learning classifiers. One reason is that CNN is better for extracting position invariant features in contrast to LSTM. Another reason is that CNN performs exceptionally well with high-dimensional feature [20].
- Second, the CNN classifier also achieves the highest performance upon the selected machine learning classifiers. One reason is that CNN transforms the non-linear and inter-dependent features into a high-dimensional plane.
- Third, although the state of the art [7], [21] suggests that Bayesian is effective in the classification of reviews, it results in performance reduction with the proposed approach. One possible reason is that some of the given non-textual features (Rev_n and Rev_r) to the classifier are inter-related, and Bayesian performs well with the independent features [15], [19]. In contrast to SVM and RF, Bayesian is not appropriate with the proposed approach.
- Fourth, we observe the slight difference in a performance comparison of SVM and RF. The evaluation of the proposed approach employing these classifiers on other datasets may influence the reported performance.

Based on the preceding analysis, we conclude that the proposed classifier is appropriate for the performance improvement of the proposed approach.

E. THREATS TO VALIDITY

A threat to external validity is that only a limited number of reviews from the selected apps are considered for the evaluation of the proposed approach. Although we observe the slight change in performance of the proposed approach among given categories, the results may not hold for other apps or adding more categories.

A threat to construct validity is that the labels in the exploited dataset could be incorrect. Maalej et al. [7] manually labeled the selected reviews that could be incorrect for different reasons [7]. As a result, such incorrect labeling could produce inaccurate results.

A threat to internal validity is that we recode the *Maalej's* with different evaluation criteria (i.e., ten-fold cross-validation). Consequently, the average results of the *Maalej's* are slightly different. To mitigate the threat, we double-check the implementation and evaluation results.

IV. RELATED WORK

It is evident that one of the major success factors for software projects is the users' involvement and their feedbacks. For example, the positive impact of user involvement is highlighted in [22], where authors suggested managing user involvement carefully otherwise it may cause more problems in contrast to benefits. Similarly, Pagano et al. [23] conducted an empirical case study for software evolution and evident that user feedback has an important piece of information for developers to not only improve the software quality but also identify the missing features. Similar to traditional requirement engineering, crowd-based requirement engineering is also a focus of researchers. In [24], [25], authors addressed the scalability issue in the case of multiple users and the significance of the tool required for the analysis of their feedback. In [26], authors focused on getting user feedback from a mobile device which includes implicit information. To develop and maintain software projects, bug repositories are considered as one of the most scrutinized tools for the collection of user feedback [27].

The analysis of reviews for app-stores (i.e., Google Play store and Apple store) got significant research attention in recent years because reviews are usually difficult to understand due to their unstructured textual information and frequency, moreover only a third of them are informative. Therefore, Ciurumelea at al. [28] developed a tool for the developers to analyze the direct and valuable feedback provided through user reviews, to better plan maintenance and evolution activities for their mobile-apps. To facilitate the reviews' analysis, Maalej et al. [7] have classified it into three major categories: i) general exploratory studies, ii) app feature extraction, and iii) reviews filtering and summarization. Furthermore, the authors provided a relationship between customer, business, and technical characteristics of mobile-apps from BlackBerry store [2]. They exploited NLP techniques with data mining to extract the correlation and trends. The results suggest that a strong correlation between mobile-app popularity and customer rating, whereas the correlation between the number of features and price is mild. Similarly, board exploratory studies for the Apple store are conducted by Hoon et al. [29] and Pagano et al. [3]. These studies identified the trends for the rating, topics discussed in reviews, quality of mobile-apps, and quality of the review. Zhang et al. [30] proposed a novel approach to automatically tag the unlabelled issue reports. This approach computes the similarity between each unlabelled issue report and user reviews related to bugs and features and also calculates the textual similarity scores between each unlabelled issue report and labeled ones.

Some studies mined user opinions and mobile-apps features from application stores. In [31], Harman et al. uses a greedy algorithm to extract the mobile-apps features from the official pages presenting the description of applications to analyze business and technical aspects of mobile-apps. In [32], Chandy et al. exploited the latent model to classify spams in the mobile app stores and categorized the reviews into malicious and normal groups. To group and extract the feature requests from the mobile-app reviews, MARA (Mobile App Review Analyzer) [33] is introduced that exploits Latent Dirichlet Allocation (LDA) and linguistic rules for identifying common topics. In [6], authors alleviated an automatic solution for topic extraction. Moreover, they used LDA for the summarization of user reviews.

Wiscom [34] analyzed user comments and ratings in three different levels. In the first level, inconsistency in reviews is discovered. Then, the reasons for liking and disliking of mobile-apps are identified. Finally, an insight into the major concerns of the users is provided. The study exploited the linear regression model to identify negative words with the help of user reviews and ratings. Furthermore, these words are applied as an input to the LDA model to find the reasons for people disliking of mobile-apps. Li et al. [1] proposed another method to analyze user satisfaction with the help of user reviews. Authors employed a predefined dictionary to match words or phrases of user reviews. App Review Mining (AR-Miner) [35] approach is introduced that extracts most informative user reviews by using Naive Bayes. The approach first removes irrelevant and noisy reviews and groups informative reviews with the help of topic modeling. Then, it uses a ranking scheme to prioritize informative reviews. Finally, it presents the most informative reviews using an intuitive visualization approach.

An automatic approach for the classification of different software artifacts has gained significant research attention. In [8], authors exploited machine learning with linguistic rules to classify user reviews into a taxonomy. This taxonomy is created by analyzing developers' emails. Bacchelli et al. [9] leveraged a natural language parser and Naive Bayes to classify useful information from developers' emails. To classify the structured and unstructured data, Zhou et al. [10] also leveraged the machine learning techniques. Martens and Maleej [36] exploited a machine learning classifier to classify the fake reviews and achieved a recall of 91% and an AUC/ROC value of 98%. Similarly, Ekanata et al. [37] exploited Naive Bayes, Support Vector Machine, Logistic Regression, and Decision Tree for the classification of reviews. The results suggest that Logistic Regression provides the best f-measure of 85% when unigram,

sentence length, and sentiment score are combined.

For the sentiment analysis, Erik Cambria [38] reported that automatically capturing the sentiments about social events, political movements, marketing campaigns, and product preferences of general public has raised interest in the scientific community and business world for the exciting open challenges, the remarkable fallouts in marketing, and financial market prediction. Later, Amir and Erik [39] explored the potential of a novel semi-supervised learning model based on the combined use of random projection scaling as part of a vector space model, and support vector machines to perform reasoning on a knowledge base. To this end, they combined a graph representation of commonsense with a linguistic resource for the lexical representation of affect. The evaluation results suggest a significant improvement in tasks such as emotion recognition and polarity detection, and propose a way for the development of semi-supervised learning approaches to big social data analytics. Furthermore, Wang et al. [40] proposed an automatic method for the construction of the domain-specific sentiment lexicon to avoid sentimental ambiguity. It incorporates the sentiment information not only from the existing lexicons but also from the corpus. They exploited an improved TF-IDF to calculate the sentiment of words. The evaluation results suggest that constructed lexicon improves the sentimental ambiguity and outperforms the state of the art approaches. Moreover, Ma et al. [41] proposed a novel solution for aspect-based sentiment analysis by exploiting commonsense knowledge. They incorporated the commonsense knowledge of sentiment related concepts in end-to-end training of LSTM network that outperforms the state of the art methods.

Based on the preceding literature analysis, we conclude that a number of researches have been proposed for the classification of reviews. However, *DCAR* differs in that it does not only leverage the deep learning classifier but also employed the statistical information of reviewers and mobile-apps. Compared to the state of the art (discussed in this section), the additional features (i.e., statistical information of reviewers) make *DCAR* different from off-the-shelf text classification approaches.

V. CONCLUSIONS AND FUTURE WORK

Automated classification of reviews from various apps is highly desirable. In this paper, we propose a deep learning based approach for the classification of reviews. Compared to existing approaches, the proposed approach does not only leverage the statistics of reviews as non-textual features that have not been employed but also exploits a deep learning technique to classify reviews. The results of the ten-fold cross-validation evaluation on real-world reviews indicate that the proposed approach significantly surpasses the state of the art.

The employed additional features i.e., statistics of reviews suggest that these features are appropriate for the classification of review. However, temporal and location-based features are not yet examined, i.e., reviews from different geographical regions/cultures may cause their sentiment. In future, it could a possible research direction to improve the results of the review classification. Moreover, it could be interesting to evaluate the related approaches with a larger dataset of real-world reviews.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No. U1813222), Tianjin Natural Science Foundation (No. 18JCYBJC16500) and Key Research and Development Project from Hebei Province (No. 19210404D).

REFERENCES

[1] H. Li, L. Zhang, L. Zhang, and J. Shen, "A user satisfaction analysis approach for software evolution," in 2010 IEEE International Conference on Progress in Informatics and Computing, vol. 2. IEEE, 2010, pp. 1093–1097.

[2] A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "App store analysis: Mining app stores for relationships between customer, business and technical characteristics," RN, vol. 14, no. 10, 2014.

[3] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in 2013 21st IEEE international requirements engineering conference (RE). IEEE, 2013, pp. 125–134.

[4] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in 2014 IEEE 22nd International Requirements Engineering Conference (RE), Aug 2014, pp. 153–162.

[5] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in 2013 10th Working Conference on Mining Software Repositories (MSR), May 2013, pp. 41–44.

[6] L. V. G. Carreño and K. Winbladh, "Analysis of user comments: an approach for software requirements evolution," in 2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013, pp. 582–591.

[7] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," Requirements Engineering, vol. 21, no. 3, pp. 311–331, 2016.

[8] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "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). IEEE, 2015, pp. 281–290.

[9] A. Bacchelli, T. Dal Sasso, M. D'Ambros, and M. Lanza, "Content classification of development emails," in 2012 34th International Conference on Software Engineering (ICSE). IEEE, 2012, pp. 375–385.

[10] Y. Zhou, Y. Tong, R. Gu, and H. Gall, "Combining text mining and data mining for bug report classification," Journal of Software: Evolution and Process, vol. 28, no. 3, pp. 150–176, 2016.

[11] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement? a text-based approach to classify change requests," in Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds, ser. CASCON âĂŹ08. New York, NY, USA: Association for Computing Machinery, 2008. [Online]. Available: https://doi.org/10.1145/1463788.1463819

[12] S. Bird, "Nltk: The natural language toolkit," in In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics, 2002.

[13] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, ser. NIPS'13. USA: Curran Associates Inc., 2013, pp. 3111–3119. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999792.2999959

[14] Q. Umer, H. Liu, and Y. Sultan, "Emotion based automated priority prediction for bug reports," IEEE Access, vol. 6, pp. 35 743–35 752, 2018.

[15] Q. Umer, H. Liu, and Y. Sultan, "Sentiment based approval prediction for enhancement reports," Journal of Systems and Software, vol. 155, pp. 57 – 69, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121219301104

[16] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, "Sentiment polarity detection for software development," Empirical Softw. Engg., vol. 23, no. 3, pp. 1352–1382, Jun. 2018. [Online]. Available: https://doi.org/10.1007/s10664-017-9546-9

[17] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," in in Proc. of LREC, 2010.

[18] Q. Umer, H. Liu, and I. Illahi, "Cnn-based automatic prioritization of bug reports," IEEE Transactions on Reliability, pp. 1–14, 2019.

[19] W. Y. Ramay, Q. Umer, X. C. Yin, C. Zhu, and I. Illahi, "Deep neural network-based severity prediction of bug reports," IEEE Access, vol. 7, pp. 46 846–46 857, 2019.

[20] B. Wang, "Disconnected recurrent neural networks for text categorization," in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 2311–2320. [Online]. Available: https://www.aclweb.org/anthology/P18-1215

[21] J. Hellerstein, T. Jayram, and I. Rish, "Recognizing end-user transactions in performance management." 07 2000, pp. 596–602.

[22] M. Bano and D. Zowghi, "A systematic review on the relationship between user involvement and system success," Information and Software Technology, vol. 58, pp. 148–169, 2015.

[23] D. Pagano and B. Brügge, "User involvement in software evolution practice: a case study," in 2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013, pp. 953–962.

[24] E. C. Groen, J. Doerr, and S. Adam, "Towards crowd-based requirements engineering a research preview," in International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2015, pp. 247–253.

[25] T. Johann and W. Maalej, "Democratic mass participation of users in requirements engineering?" in 2015 IEEE 23rd international requirements engineering conference (RE). IEEE, 2015, pp. 256–261.

[26] N. Seyff, F. Graf, and N. Maiden, "Using mobile re tools to give end-users their own voice," in 2010 18th IEEE International Requirements Engineering Conference. IEEE, 2010, pp. 37–46.

[27] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "What makes a good bug report?" in Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, 2008, pp. 308–318.

[28] A. Ciurumelea, S. Panichella, and H. C. Gall, "Automated user reviews analyser," in Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, ser. ICSE âĂŹ18. New York, NY, USA: Association for Computing Machinery, 2018, p. 317âĂŞ318. [Online]. Available: https://doi.org/10.1145/3183440.3194988

[29] L. Hoon, R. Vasa, J.-G. Schneider, J. Grundy et al., "An analysis of the mobile app review landscape: trends and implications," Faculty of Information and Communication Technologies, Swinburne University of Technology, Tech. Rep, 2013.

[30] T. Zhang, H. Li, Z. Xu, J. Liu, R. Huang, and Y. Shen, "Labelling issue reports in mobile apps," IET Software, vol. 13, no. 6, pp. 528–542, 2019.

[31] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: Msr for app stores," in 2012 9th IEEE working conference on mining software repositories (MSR). IEEE, 2012, pp. 108–111.

[32] R. Chandy and H. Gu, "Identifying spam in the ios app store," in Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality, 2012, pp. 56–59.

[33] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in 2013 10th working conference on mining software repositories (MSR). IEEE, 2013, pp. 41–44.

[34] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "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, 2013, pp. 1276–1284.

[35] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang, "Ar-miner: mining informative reviews for developers from mobile app marketplace," in Proceedings of the 36th international conference on software engineering, 2014, pp. 767–778.

[36] D. Martens and W. Maalej, "Towards understanding and detecting fake reviews in app stores," Empirical Software Engineering, vol. 24, no. 6, p. 3316âĂŞ3355, May 2019. [Online]. Available: http://dx.doi.org/10.1007/s10664-019-09706-9

[37] Y. Ekanata and I. Budi, "Mobile application review classification for the indonesian language using machine learning approach," in 2018 4th International Conference on Computer and Technology Applications (ICCTA), May 2018, pp. 117–121.

[38] E. Cambria, "Affective computing and sentiment analysis," IEEE Intelligent Systems, vol. 31, no. 2, pp. 102–107, 2016.

[39] A. Hussain and E. Cambria, "Semi-supervised learning for big social data analysis," Neurocomputing, vol. 275, pp. 1662 – 1673, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217316363

[40] Y. Wang, F. Yin, J. Liu, and M. Tosato, "Automatic construction of domain sentiment lexicon for semantic disambiguation," Multimedia Tools and Applications, 05 2020.

[41] Y. Ma, H. Peng, T. Khan, E. Cambria, and A. Hussain, "Sentic lstm: a hybrid network for targeted aspect-based sentiment analysis," Cognitive Computation, pp. 1–12, 03 2018.