Neural Sentiment Analysis of User Reviews to Predict User Ratings

Bahar Gezici[1], Necva Bölücü[2], Ayça Tarhan[1], and Burcu Can[2]

Computer Engineering Department
Hacettepe University, Beytepe, Ankara, Turkey
{bahargezici, necva, atarhan, burcucan}@cs.hacettepe.edu.tr
[1]Software Engineering Research Group (HUSE)
[2]Natural Language Processing Research Group

***Abstract*—The significance of user satisfaction is increasing in the competitive open source software (OSS) market. Application stores let users send their feedbacks for applications, which are in the form of user reviews or ratings. Developers are informed about bugs or any additional requirements with the help of this feedback and use it to increase the quality of the software. Moreover, potential users rely on this information as a success indicator to decide downloading the applications. Since it is usually costly to read all the reviews and evaluate their content, the ratings are taken as the base for the assessment. This makes the consistency of the contents with the ratings of the reviews important for healthy evaluation of the applications. In this study, we use recurrent neural networks to analyze the reviews automatically, and thereby predict the user ratings based on the reviews. We apply transfer learning from a huge volume, gold dataset of Amazon Customer Reviews. We evaluate the performance of our model on three mobile OSS applications in the Google Play Store and compare the predicted ratings and the original ratings of the users. Eventually, the predicted ratings have an accuracy of 87.61% compared to the original ratings of the users, which seems promising to obtain the ratings from the reviews especially if the former is absent or its consistency with the reviews is weak.**

***Keywords*- Open Source, OSS, Transfer Learning, Sentiment Analysis, Deep Learning, User Satisfaction, Software Quality**

I. INTRODUCTION

User satisfaction is a frequently used dimension in evaluating and selecting Open Source Software (OSS) [5, 28]. It is measured by analyzing user reviews and ratings that are allowed in application markets to collect feedback from the users regarding various software releases. This feedback enables to assess and estimate the quality and success of an OSS in time [5, 8].

User reviews include information that is useful for analysts and application designers, such as user requirements, bug reports, feature requests, and documents of user experiences with specific application features. In addition, users rate an application by assigning a numerical value which ranges from a minimum value denoting less satisfied to a maximum value denoting very satisfied. Low-rated applications negatively affect the perceived quality of the application, thus affecting the popularity of the application and even the revenue adversely [5].

There is a number of studies that rely on user reviews or ratings to measure and analyze the quality of software [5, 7, 21]. Although it is reasonable to do that, it is usually costly to read all the reviews and evaluate their content so that the ratings are taken as the base for the analysis. This makes the consistency of the contents with the ratings of the reviews important for healthy evaluation of the applications. Also, in some cases, users are unwilling to rate the applications that they download from the stores, which leads to missing or even misleading ratings.

Due to challenges mentioned above, some studies automate the analysis process by applying sentiment analysis as one of the most known techniques in Natural Language Processing (NLP). Sentiment analysis has been applied in software engineering to automate the analysis of issue comments [2, 11], e-mail contents [6, 27] and forum posts [19]. However, applying sentiment analysis on incomplete or inconsistent datasets are likely to produce inefficient results [23]. Deep learning has recently emerged as a powerful machine learning technique and has been used in many application domains ranging from computer vision to software engineering. When working with datasets of low quality such as noise data, feature distribution differences or imbalance data etc., it can be supported by transfer learning where knowledge is transferred from a related task for improvement of learning in a new task [1, 29]. Therefore, combining transfer learning with deep learning for the sentiment analysis of user reviews might be a solution to predict user ratings in a healthy way.

Accordingly, the contribution of this paper is twofold. First, we present a deep learning model that predicts user ratings in order to cope with the challenge of inconsistent and insufficient ratings given by the users. We then apply the model on metric data of three mobile OSS applications from Google Play market, and evaluate its performance with respect to original user ratings of applications. Second, we use Success Index (SI) for the OSS applications in accordance to another study of two authors [8] and compare the correlations of SI with both the original and predicted user ratings in order to observe any variation on the relations.

The remaining of this paper is structured as follows: Section II addresses the background and related work essential to understand the study. Section III describes the research questions and design. Section IV provides details of prediction model. Section V presents the experimental details, and Section VI gives the experimental results obtained from the proposed model. Finally, Section VII closes the paper with conclusions and future work.

II. BACKGROUND & RELATED WORK

A. OSS Quality & Success

The success of an open source software in the application store is based on a number of complex factors such as originality and marketing strategy. Paying attention to a well-coded software impacts the success of an application as a

final product. Some studies are investigating the impact of the quality of the source code in the market success of a mobile application while the market success could be defined in terms of the feedback provided by the application store, such as the number of e-mails, number of downloads, user ratings, and other metrics.

As an example, Corral et al. [5] proposed a study to determine whether there is a relationship between product quality and market success and they found that there is a significant impact between software quality and success. Another related study with this context belongs to Gezici et al. [8], where the market success as a demonstration of external quality in the community-based dimension was evaluated from multiple perspectives. Based on their model that consists of six related metrics, a Success Index (SI) was formulated based on statistical correlations between the metrics. According to the results of Spearman correlation analysis, there is a significant correlation between some internal quality attributes and community-based success (SI).

B. Sentiment Analysis

Sentiment analysis is the task of assigning a quantitative value to a text with respect to the opinion, sentiment or emotion towards various entities such as products and organizations [31]. In the last years, interest in sentiment analysis in software engineering domain has risen significantly. There are studies that analyse users' satisfaction [12] and developers' emotions [17] automatically by applying sentiment analysis. Several other studies are summarized below.

Lin et al. [16] build a software library recommender that utilizes developers' opinions mined from Stack Overflow.

Kaur et al. [13] introduce an approach that categorizes texts on the basis of their sentiments manually by adapting the study of Murgia et al. [18]. They collect their supervised dataset (Apache JIRA) for this study.

There are some studies that analyze user reviews to evaluate mobile applications through their evolution [10], [12], [21]. Goul et al. [10] propose a study that observes 5,000 reviews by applying sentiment analysis tool. Sinha et al. [25] study similar topic on projects that last for 7 years.

Panichella et al. [21] also perform sentiment analysis by using Naive Bayes classifier on users' review.

C. Deep Learning

Deep learning is the application of artificial neural networks that aims to discover better representations of the inputs provided during training. There are various types of neural networks proposed in the literature. One of them is Recurrent Neural Networks (RNNs). RNNs are specially designed for input sequences, which makes it useful for Natural Language Processing tasks.The LSTMs are an extension of Recurrent Neural Networks, which basically extend the internal memory.

In literature, there are various studies that use deep learning models applied in software engineering domain as surveyed in [15] such as bug report summarization [14] and software defect prediction [26]. In this context, there are also studies that review rating prediction by using deep learning models. For example, Zheng et al. [32] introduce a Neural Network based model, named Deep Cooperative Neural Networks (DeepCoNN), for modelling items and users together using reviews for rating prediction problems. Cheng et al. [4] also represent textual review information with ratings to tackle cold-start, non-transparency, and suboptimal recommendation for local users or items limitations.

Seo et al. [24] present a Convolutional Neural Network (CNN) model with attention mechanism for users and items to predict the rating values of a user for an item.

D. Transfer Learning

Transfer learning, proposed by Pan and Yang [20] in 2010, is a popular approach in machine learning. Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims to improve the learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$.

Al-Stouhi and Reddy [1] propose a novel boosting-based instance transfer classifier with a label-dependent update mechanism. This mechanism is used for imbalance data with samples from an auxiliary domain to improve classification. The authors apply this mechanism to healthcare and text classification applications.

In another study, Cao et al. [3] present Transfer Component Analysis Neural Network (TCANN) model for software defect prediction by using transfer learning based neural network.

There is no study that solves the imbalance data problem with transfer learning for quality rating prediction with deep learning models. Our study provides a novel contribution to literature by doing this.

III. RESEARCH DESIGN

A. Research Questions

This study aims to automate the analysis of user reviews of OSS (by using LSTM and transfer learning) in order to predict the user ratings, and investigates any variation in the correlation between the predicted user ratings and the perceived success of OSS. Accordingly, the following Research Questions (RQs) are raised in this study:

- **RQ 1.** Can user ratings of OSS be predicted accurately by an LSTM model?
- **RQ 2.** Does the correlation of SI with user ratings vary when predicted?

To answer these research questions, we use two metrics as described below.

- **Metric 1. User Ratings (UR):** This metric serves as an indicator of how satisfied the user is with the application. It is the average value calculated from all votes given by users. User ratings in application stores are obtained by dividing the total values of all ratings by the number of reviewers, which shows an average of how satisfied the user is by the application.
- **Metric 2. Success Index (SI):** It is a metric that indicates the community-based quality (perceived success) of an application. It is applied by a statistical correlation method and calculated by considering the metrics addressing three success dimensions (Software Use, User Satisfaction, and Development Community Service Quality) [8].

According to the study [8], SI was formulated consisting of six related metrics: user ratings (UR), number of reviews (NOR), number of commits (CN), number of developers (NOD), number of feedbacks (NOF) and time between

releases (TBR). Each metric was examined according to the status of a meaningful relation with other metrics as shown in Table I. After determining the directions of the relationships between the pairs of metrics, an equation for SI was developed taking these relations into consideration. Then, the metrics with meaningless relations, according to the significance status of the SI with each metric, were removed from the equation and a new equation was obtained. Accordingly, SI was calculated as:

TABLE I
SPEARMAN CORRELATION COEFFICIENT BETWEEN METRICS (r_s)[1]

Metric	NOR	UR	NOF	NOD	CN	TBR
NOR	1	0.113	-0.048	0.291	-0.149	-0.732
UR	0.113	1	0.094	-0.257	-0.303	0.281
NOF	-0.048	0.094	1	0.132	0.100	0.117
NOD	0.291	-0.257	0.132	1	0.227	-0.304
CN	-0.149	-0.303	0.100	0.227	1	-0.155
TBR	-0.732	0.281	0.117	-0.304	-0.155	1

[1]The gray areas indicate Spearman correlation coefficients that have a significant relationship [8]

$$SI = \frac{(NOR \times UR \times NOF \times NOD)}{(CN \times TBR)}$$

B. Selection of Open Source Software

This study is a following of a recent study [8] that has two drawbacks with respect to extraction of reviews and consistency of reviews. Therefore, we selected the same three OSS applications in there according to following criteria:

- Application platform similarity. The applications should be developed for similar mobile platforms in order to reduce the potential impacts of platform disparity.
- Application functionality similarity. Applications within each case study must have similar functionality, in order to reduce the possible effects of different functionality.

We use the reviews and the ratings of the same OSS applications as a single source and apply the proposed model on this file. The reason we selected the same OSS applications is to check whether the correlation of SI with user ratings changes when predicted.

C. Metrics for Performance Evaluation

The metrics used to evaluate the results of the rating prediction are given Table II.

TABLE II
EVALUATION METRICS

Evaluation Metric	Formula
Accuracy	$A = \frac{R_s}{R_w}$
Precision	$precision = \frac{true\_positives}{true\_positives + false\_positives}$
Recall	$recall = \frac{true\_positives}{true\_positives + false\_negatives}$
F1-score	$F1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

Spearman Correlation Analysis [30] is a statistical method used to test the relationship between two variables, or analyze the relationship of one variable to two or more variables, if any, and to measure the degree of this relationship. In this analysis, r_s expresses the correlation coefficient between two or more variables and ρ indicates the significance of Spearman correlation. The ρ value should be less than 0.05 ($\rho \leq 0.05$) for the significant correlation.

IV. ANALYSIS METHOD

Within the scope of this study, we automatically predict the ratings based on user reviews by using transfer learning. We transfer information from a huge volume dataset (Amazon Customer Reviews Dataset) to our dataset explained in V-A. To perform this, we use NLP and Machine Learning techniques. We adopt Long Short Term Memory networks (LSTMs) in this study.

Figure 1 shows an overview of the analysis method. First, we collect the user reviews for the selected applications and extract the reviews, original ratings, and the versions of the applications from the App store. Then, we use deep neural networks to avoid any handcrafted features. We use Amazon Customer Reviews Dataset to train our model on a large dataset. Eventually, we use a transferred model to predict the ratings.

We calculated accuracy of predicted ratings extracted dataset. Then we analyze the correlation between SI and UR for both the predicted ratings and the original ratings, respectively. Finally, we use Spearman Correlation Analysis to test the relationship between predicted UR-SI and original UR-SI, and compare the results to see if there is a significant variation in the correlation.

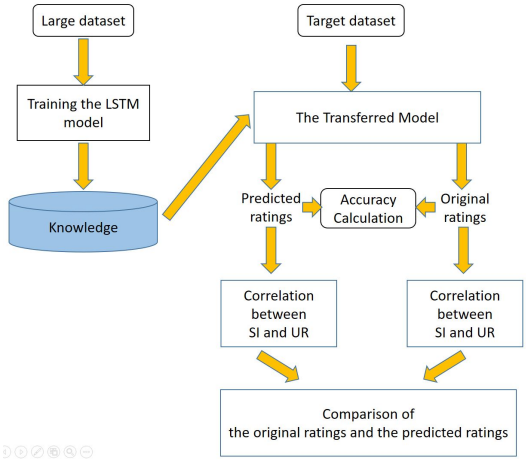

Fig. 1. The analysis method

A. Preprocessing Step

Before applying the proposed model, preprocessing step is applied to clean and prepare datasets for the task. To convert input data as inputs of the proposed model, we do some preprocessing on both transfer learning dataset (Amazon Customer Review Dataset) and extracted dataset.

As the first step of preprocessing, we convert all words to lowercase to decrease the number of unique words (e.g. “However” to “however”) and remove punctuation. Finally, each word is mapped to their corresponding word embedding taken from pretrained word embedding. Each word is represented in a low dimensional space that defines both syntactic and semantic features of words. Since we are interested in opinion, we aim capture emotions using the words embeddings.

B. The Proposed Model

In this work, we propose an LSTM model that takes sentences in a training set D as an input sequence to output the sentiment analysis of user reviews. Given a sentence,

proposed model computes a score for each sentiment label (ranging from "1" to "5"). The architecture of our network is given in Figure 2.

Given a sentence x with n words $\{w_1, w_2, \cdots, w_n\}$, we first map every word w_i to a word embedding that are pretrained by GloVe [22]. Then, we use an LSTM that receives a sequence of word embeddings and generates a prediction. The output of each LSTM unit is given to sigmoid function. We use only the last output of the sequence. Our network is trained by minimizing the negative likelihood over the training set D.

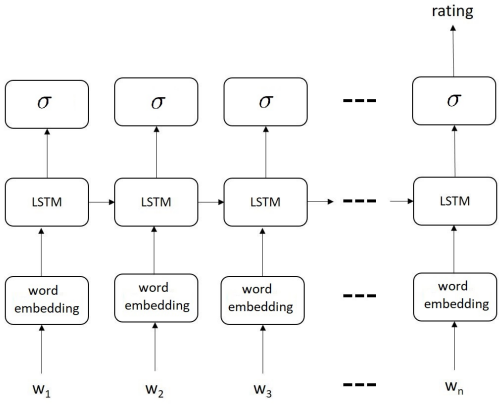

Fig. 2. Illustration of the neural model for the rating prediction

V. EXPERIMENTAL DETAILS

A. Dataset & Hyperparameters for Transfer Learning

Transfer Learning Dataset: Amazon Customer Reviews Dataset is a collection of customer reviews and metadata written on *Amazon.com* from 1995 until 2015. The dataset contains four million reviews and star ratings. Amazon Customer Reviews dataset is balance and also benchmark dataset that is used as dataset in transfer learning for sentiment analysis task [9].

This dataset is preferred due to the similar structure with our extracted dataset. They both contain user reviews and ratings about products and apps. Also, the reviews are comply with each other. The example reviews for two dataset are given in Table III.

When we analyze the datasets, the reviews are compatible. For more detailed analysis we extracted common words used in the reviews for the datasets. It is seen that common adjectives and phrases in the reviews are similar for Amazon Customer Reviews Dataset and our extracted dataset.

Therefore, it is assumed that our proposed model trained on Amazon Customer Reviews dataset for prediction of ratings of selected OSS application reviews will not have a negative effect on the sentiment analysis.

Evaluated Hyperparameters: The hyperparameters of the network from which we obtain the best results, are given in Table IV.

B. Data Collection for Community-based Metrics

We collected and preprocessed the user reviews and corresponding original ratings that are scaled from 1 to 5, where 1 is the lowest (negative) and 5 is the highest (positive) rating. Additionally, we obtained the time information between releases to classify the ratings with respect to its versions for 46 different releases of the three OSS applications (9 releases of Keepassdroid, 4 releases of UPM and 33 releases of PasswdSafe. Detailed information of our dataset is given in Table V.

In total, we obtained 1557 reviews from the extracted dataset for the three applications. There are 976 reviews with a user rating of "5", 246 reviews with a user rating of "4", 14 reviews for a user rating of "3", 96 reviews for a user rating of "2", and 125 reviews with a user rating of "1".

For the chosen three applications and the transfer learning dataset (Amazon Customer Reviews Dataset), we made a barplot for the distribution of the ratings. All the rating distributions for the three applications and the transfer learning dataset are imbalanced. Interestingly, people tend to publish their opinions on applications which are positive. Although, some flaunt negative reviews exist for the three applications, the number of these reviews is much less. In this regard, negative reviews are more valuable than positive reviews for these applications, and transfer learning is expected to elicit negative reviews and to better represent them in the ratings.

VI. RESULTS & DISCUSSION

A. Experimental Results

Our evaluation goal was twofold. First, we evaluated the predicted ratings. Experimental results are given in Table VI. The results indicates that predicted ratings converge to the original ratings. Second, we used Spearman's Correlation Analysis to observe whether there is a significant relationship between SI and UR for both predicted and original ratings, and any variation in the correlations.

The confusion matrix for the results is given in Figure 3. It is clearly seen that due to the number of reviews with "5" in the training set, the model tends to label reviews with rating "5". However, the model predicts the ratings accurately in general.

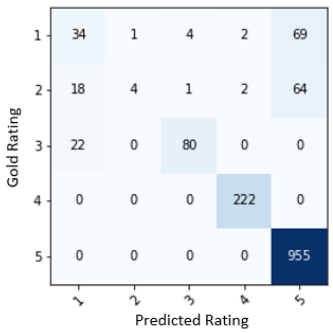

Fig. 3. Confusion Matrix of the model

To observe relationship between **SI** and **UR** for both predicted and original ratings, Spearman Correlation Analysis results are given in Table VII.

All community-based metrics (UR) belonging to a total of 46 releases of Keepassdroid, UPM and PasswdSafe were subject to this test. After the test, we observed that there is a significant relationship between UR and SI for both predicted and original ratings based on ρ parameter. In terms of r_s parameter, the correlation between **Predicted UR** and **Original SI** is higher than the correlation between **Original UR** and **Original SI**, though with a relatively lower but still strong significance.

TABLE III
EXAMPLE REVIEWS FROM AMAZON CUSTOMER REVIEWS DATASET AND EXTRACTED DATASET

Amazon Customer Reviews Dataset	Extracted Dataset
I like this app better than some free fall apps	i like this app i combined with dropbox sync it works
I love this app I wish it was real	love this app been using it for a while now couldn't live without it
great app	great app
I can now watch videos on my phone! Love this app	Love this app
This works great on your Kindle or Android	works great

TABLE IV
HYPERPARAMETERS USED IN THE EXPERIMENTS

Pre-trained Word Embeddings	GloVe 42B 300d
Number of LSTM-Layers	1
Optimizer	Adam
Dropout	0.4
Number of Recurrent Units	300
Mini-batch Size	128
Backend	Tensorflow

TABLE V
OVERVIEW OF THE ANALZED OSS APPLICATIONS

App	Platform	Lines of code	#Releases	#Reviews
Keepassdroid	Java	4226	127	907
UPM	Java	1345	16	249
PasswdSafe	Java	5728	92	419

TABLE VI
EXPERIMENTAL RESULTS OF THE MODEL

	Precision	Recall	F1-score	Accuracy
Model	81.21	62.76	64.72	87.61

TABLE VII
RESULTS OF SPEARMAN CORRELATION ANALYSIS

			Original SI
Spearman Correlation Analysis	**Predicted UR**	r_s	0.204
		ρ	0.041
		N	46
	Original UR	r_s	0.068
		ρ	0.005
		N	46

As a result, we can conclude that we do not really need to take ratings of users to calculate the SI or the correlation of SI and UR. We can automatically extract our ratings with the RNN model and use them for the evaluation. The proposed method may provide us flexibility to use various reviews of applications that do not include ratings in application stores.

B. Validity Threads

The selected applications are open source applications and they are accessed from trusted data repositories such as GitHub and Google Play market. Measurements are also automatically performed using the Xlstat software and relations between metrics are evaluated considering Spearman Correlation Analysis. At this point, the emergence of human-induced mistakes was greatly hampered. Dataset selection could be seen as a constraint on internal validity, since another dataset from Amazon was considered for training and the model was then tested with real dataset. However, this can be considered a widespread practice in data science applications.

The method was employed for 1) predicting ratings from user reviews with machine learning models, 2) analyzing the relation of community-based quality (SI) in terms of the existing metric (UR) throughout the releases of the mobile applications; and therefore provides an initial basis for repeating (generalizing) the steps in further similar empirical studies. Also, the proposed RNN model is repeatable and reusable for datasets from the same or different domains.

VII. CONCLUSION

This study presents a method to predict user ratings from user reviews. The proposed method uses a huge volume dataset with the same ranked ratings to learn the model and predicts ratings based on this oracle model. We obtained 87.61% accuracy for the rating prediction task. The correlation of SI with UR was significant for both predicted and original ratings. Moreover, SI was better correlated with predicted UR than the original one.

As a future study, it is planned to repeat this method for different OSS and also to measure community-based success of different applications with different metrics (e.g. feedback in projects' e-mail lists or in discussion groups).

REFERENCES

[1] Samir Al-Stouhi and Chandan K Reddy. Transfer learning for class imbalance problems with inadequate data. *Knowledge and information systems*, 48(1):201–228, 2016.

[2] Fabio Calefato and Filippo Lanubile. Affective trust as a predictor of successful collaboration in distributed software projects. In *2016 IEEE/ACM 1st International Workshop on Emotional Awareness in Software Engineering (SEmotion)*, pages 3–5. IEEE, 2016.

[3] Qimeng Cao, Qing Sun, Qinghua Cao, and Huobin Tan. Software defect prediction via transfer learning based neural network. In *2015 First International Conference on Reliability Systems Engineering (ICRSE)*, pages 1–10. IEEE, 2015.

[4] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan Kankanhalli. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 639–648. International World Wide Web Conferences Steering Committee, 2018.

[5] Luis Corral and Ilenia Fronza. Better code for better apps: A study on source code quality and market success of android applications. In *2015 2nd ACM International Conference on Mobile Software Engineering and Systems*, pages 22–32. IEEE, 2015.

[6] David Garcia, Marcelo Serrano Zanetti, and Frank Schweitzer. The role of emotions in contributors activity: A case study on the gentoo community. In *2013 International Conference on Cloud and Green Computing*, pages 410–417. IEEE, 2013.

[7] Bahar Gezici. Mobil uygulamaların evriminde kalitenin gelişimi. Master's thesis, Fen Bilimleri Enstitüsü, 2018.

[8] Bahar Gezici, Ayça Tarhan, and Oumout Chouseinoglou. Internal and external quality in the evolution of mobile software: An exploratory study in open-source market. *Information and Software Technology*, 2019.

[9] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011.

[10] Michael Goul, Olivera Marjanovic, Susan Baxley, and Karen Vizecky. Managing the enterprise business intelligence app store: Sentiment analysis supported requirements engineering. In *2012 45th Hawaii International Conference on System Sciences*, pages 4168–4177. IEEE, 2012.

[11] Emitza Guzman, David Azócar, and Yang Li. Sentiment analysis of commit comments in github: An empirical study. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 352–355. ACM, 2014.

[12] Emitza Guzman and Walid Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)*, pages 153–162. IEEE, 2014.

[13] Arvinder Kaur, Amrit Pal Singh, Guneet Singh Dhillon, and Divesh Bisht. Emotion mining and sentiment analysis in software engineering domain. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1170–1173. IEEE, 2018.

[14] Xiaochen Li, He Jiang, Dong Liu, Zhilei Ren, and Ge Li. Unsupervised deep bug report summarization. In *Proceedings of the 26th Conference on Program Comprehension*, pages 144–155. ACM, 2018.

[15] Xiaochen Li, He Jiang, Zhilei Ren, Ge Li, and Jingxuan Zhang. Deep learning in software engineering. *arXiv preprint arXiv:1805.04825*, 2018.

[16] Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, Michele Lanza, and Rocco Oliveto. Sentiment analysis for software engineering: How far can we go? 2018.

[17] Ivy LB Liu, Christy MK Cheung, and Matthew KO Lee. User satisfaction with microblogging: Information dissemination versus social networking. *Journal of the Association for Information Science and Technology*, 67(1):56–70, 2016.

[18] Alessandro Murgia, Parastou Tourani, Bram Adams, and Marco Ortu. Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In *Proceedings of the 11th working conference on mining software repositories*, pages 262–271. ACM, 2014.

[19] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. Towards discovering the role of emotions in stack overflow. In *Proceedings of the 6th international workshop on social software engineering*, pages 33–36. ACM, 2014.

[20] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[21] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A Visaggio, Gerardo Canfora, and Harald 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)*, pages 281–290. IEEE, 2015.

[22] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[23] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. Evaluation datasets for twitter sentiment analysis. *Emotion and Sentiment in Social and Expressive Media*, page 9, 2013.

[24] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Representation learning of users and items for review rating prediction using attention-based convolutional neural network. In *3rd International Workshop on Machine Learning Methods for Recommender Systems (MLRec)(SDM17)*, 2017.

[25] Vinayak Sinha, Alina Lazar, and Bonita Sharif. Analyzing developer sentiment in commit logs. In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 520–523. ACM, 2016.

[26] Haonan Tong, Bin Liu, and Shihai Wang. Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning. *Information and Software Technology*, 96:94–111, 2018.

[27] Parastou Tourani, Yujuan Jiang, and Bram Adams. Monitoring sentiment in open source mailing lists: Exploratory study on the apache ecosystem. In *Proceedings of 24th annual international conference on computer science and software engineering*, pages 34–44. IBM Corp., 2014.

[28] Nebi Yılmaz. Açık kaynak yazılımlarda bakım yapılabilirliği ve güvenirliği ölçmek için iki boyutlu değerlendirme metodu. Master's thesis, Fen Bilimleri Enstitüsü, 2017.

[29] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[30] Jerrold H Zar. Significance testing of the spearman rank correlation coefficient. *Journal of the American Statistical Association*, 67(339):578–580, 1972.

[31] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

[32] Lei Zheng, Vahid Noroozi, and Philip S Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 425–434. ACM, 2017.