Automated Bugs Identification Through Early Access Game Review Analytics On Game Distribution Platforms

Ali Shahbaz
Department of Software Engineering
National University of Computer and
Emerging Sciences (NUCES)
alishahbazbal1234@gmail.com

Dr. Khubaib Amjad Alam
Department of Software Engineering
National University of Computer and
Emerging Sciences (NUCES)
khubaib.amjad@nu.edu.pk

***Abstract*— User reviews are considered one of the most important source of information about an app and game. Game applications receive user input in the form of reviews, which can assist developers in selecting games with improved functionality and extracting relevant information such as user feedback, problems, and descriptions of user experiences linked with existing features. Recently, several research studies have been conducted to mine and categorize user-reviews into actionable software maintenance requests, including feature requests and bug reports for different software systems. However, existing literature have mostly focused on mining functional aspects and analysis of use for software systems. But for gaming industry existing literature is limited to manual analysis of mining functional aspects and analysis of use. To achieve user satisfaction and to survive in the gaming market, addressing these bugs reports to developers is necessary. Therefore, in this research we formulate this problem as a Multi-label classification problem and propose a bug classification models using RNN (recurrent neural network), LSTM (long term short term memory) and CNN (Convolutional neural network) all vary in accuracy. Representative features are used to train a model for bug classification. Use of feature extraction methods to train a classification model may lead to divergent results, which implies the need for a careful selection of these methods. Several recent studies have emphasized basic state-of-the-art taxonomies for bug classifications into different categories. Therefore, keeping in view these taxonomies this research employs these taxonomies to provide a tools which takes a user reviews dataset and then identify bugs from them and after identification, it will classify the bug into their related bug categories using classification model. We are taking GAME STEAM ENGINE as a case study to scrap gaming reviews and train model on that reviews for bugs classification.**

Key-Words: Text mining, classification, software repositories, open source software projects, triaging, feature extraction, Gaming reviews

1. Introduction

Computer games have grown in popularity throughout time, resulting in a multi-billion-dollar industry. Making game quality a crucial consideration. The majority of online gaming stores allow consumers to leave feedback on a game they purchased. Such reviews have the power to make or ruin a game. Other potential consumers frequently rely their purchasing decisions on a game's reviews. As a result, reading game evaluations can aid game makers in better comprehending user issues and continue to increase the games user-perceived quality [1].

It has gained a lot of momentum and is now a major software development industry. Platform games are back in both 2D and 3D (the developed version). A big challenge for them is that the industry lacks an automated system-level approach to gaming test. Currently, most game development organizations. The game is tested using manual or semi-automatic techniques. Such testing methods do not meet industry requirements if you need a more systematic and repeatable approach [2].

The app distribution platforms have embraced an open business approach in order to reduce the limitations for small-scale businesses and commercial app developers. Lowering the entry barriers has led to increased competition in the mobile app market [3],[4]. The manual analysis is not practical for a large number of reviews, received every day. The research on app review mining has used text mining techniques to extract useful information from the reviews. One way to find valuable information in reviews is to train an ML-based classification model to automatically group the reviews according to their content [5].

Developers have difficulty writing comprehensive tests, because games can have a significantly large number of possible user interactions when compared to other types of software. As a result, many games will be released with it undiscovered bugs that only reveal themselves to the customer start playing the game [6].

This difficulty can often lead to players experiencing failure, which negatively affects the game experience [7]. This challenge draws attention to a more comprehensive analysis of reviews to identify issues and requirements, given the potential value that the text of the review means.

In this paper we present an automated tool which takes a user review dataset and then identify bugs from them and after identification it will classify the bug into their related bug categories. The goal of this thesis was to automate the extraction of gaming bugs from steam engine game reviews, train classification models to categorize user reviews under different labels, and make the bug categorization process more efficient.

2. Problem Formulation

Mining and analyzing user game reviews has emerged as an important challenge for game developers. The excessive amount of user ratings makes it difficult to analyze and manually interpret the ratings for information extraction.

While dealing with large set of data training of classification models are performed using feature extraction [14].

The importance of these models for multi-label text classification is not clear. This paper addresses this issue and evaluates the performance of the classification model. The approach to bug classification in game reviews refers to different datasets or evaluation strategies, so the results cannot be compared. To solve this problem, this paper first evaluates the performance of basic feature extraction methods for labeled datasets and the performance of monitored deep learning models. Because reviews are in text format and contain additional information that isn't important to get as a feature of your app, this strategy risks extracting a lot of irrelevant features. To address this challenge, this task involves training a convolutional neural network model to perform validation classifications without explicit use of feature extraction methods.

2.1 literature survey

Some of the already implemented techniques for software bugs classification are:

a) D. Lin, C. P. Bezemer, Y. Zou, and A. E. Hassan, An empirical study of game reviews on the Steam platform, vol. 24, no. 1. Empirical Software Engineering" [1].

b) S. Iftikhar, M. Z. Iqbal, M. U. Khan, and W. Mahmood, "An automated model based testing approach for platform games" [2].

c) D. Lin, C. P. Bezemer, and A. E. Hassan, "Identifying gameplay videos that exhibit bugs in computer games," [3].

d) Tilmann Bruckhaus provided a technique for "Escalation Prediction to avoid escalations by predicting the defects that have high escalation risk and then by resolving them proactively" [4].

e) J. P. Zagal, A. Ladd, and T. Johnson, "Characterizing and understanding game reviews," [5].

f) L. F. S. Britto and L. D. S. Pacífico, "Evaluating Video Game Acceptance in Game Reviews using Sentiment Analysis Techniques," [6].

g) R. Deocadez, R. Harrison, and D. Rodriguez, "Preliminary study on applying Semi-Supervised Learning to app store analysis," [7].

h) L. F. S. Britto and L. D. S. Pacífico, "Evaluating Video Game Acceptance in Game Reviews using Sentiment Analysis Techniques," [8].

3. Problem Solution

This section describes the proposed system for automated classification of bugs, data used for classification and results obtained in different classification models.

3.1 Input Data

Reviews are scrapped from STEAM ENGINE GAME PLATFORM using customized web scrapper known as SCRAPY. After Pre-processing reviews are labelled into fifteen different bugs categories [1].

3.2 Model for prediction

After conducting pilot studies using multiple classification models (RNN, CNN, LSTM). LSTM have shown higher accuracy for multi-label gaming bugs classification. High-level architecture for the implementation of the proposed classification model LSTM is shown in Figure 1.

3.2.1 Pre-processing

The key stage in data mining is data pre-processing. Since the data from gaming reviews is unprocessed, it cannot be used to train the classification system directly. Because it contains unnecessary information, special symbols etc. In order to make the data useable for training purposes, it is first pre-processed. For pre-processing NLP techniques stop word removal, bag of word and lemmatization are applied before passing data for processing to model.

3.2.2 Feature Selection & Bugs Labelling

After using the "bag of words" technique on the data, a very large dimensionality was obtained in the vocabulary. The majority of these parameters have no bearing on text categorization, which lowers the classifier's effectiveness. The method of feature selection, which selects the top k phrases from the entire lexicon and reduces the dimensionality, is used to increase accuracy and efficiency.

There are a multiple feature selection techniques such as Pearson Correlation Recursive Feature Elimination

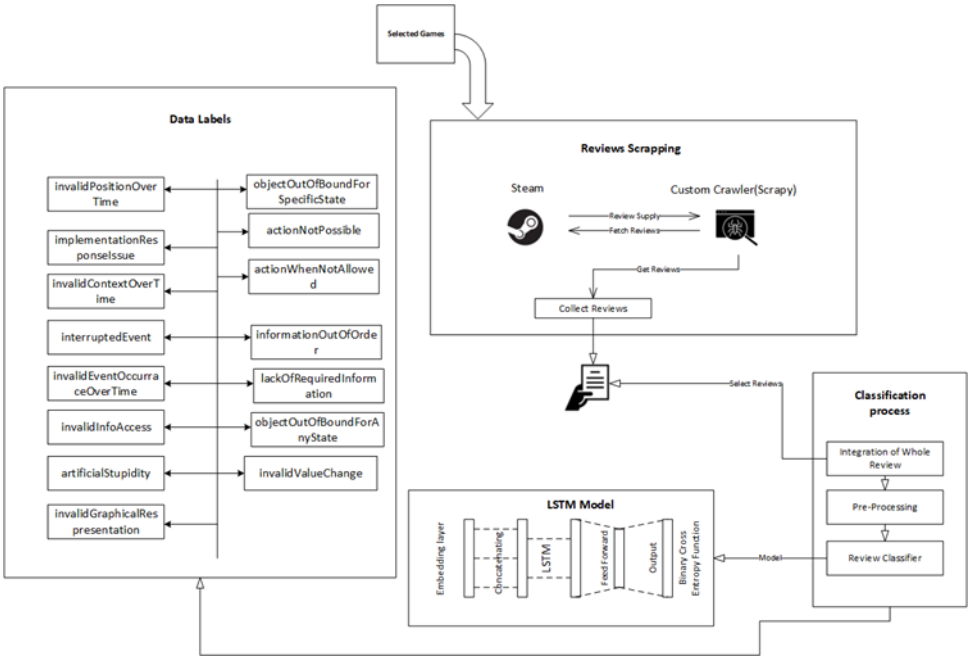

Figure 1. Architecture Diagram of Proposed Bugs Classification Model

Frequency Inverse Document Frequency (TFIDF). In this research we have used TFIDF for feature selection. Bugs are labelled into fifteen different categories (shown in figure 1) of gaming bugs by gaming experts.

3.2.3 Classifier Modeling

LSTM is a kind of recurrent neural network, but it has a better memory than traditional recurrent neural networks. LSTMs work much better due to hold over memorizing certain memory patterns.

LSTMs can have multiple hidden layers, and as you go through each layer, that information is stored and all the insignificant information in each cell is thrown away. It is implemented using python torch (machine learning) library. Multiple layers of LSTM are implemented. Sigmoid function is used as an activation function.

4. Experimental Results

Results are obtained on the basis of prediction accuracy.

Prediction Accuracy is defined as

"*Difference between the observed values and predicted values"*

For feature extraction TFIDF is used. SIGMOID is used as an activation function for the classification model. For loss function binary cross entropy function is used. The results are shown in below figures.

Figure 3,4 shows the model accuracy for each epochs. It clearly shows that prediction accuracy increases as training to testing ratio increases. Increasing the training to testing ratio although increases the prediction accuracy but execution time of algorithm and computation power increases as well.

The proposed LSTM model working is shown in figure 2. LSTM receives five arguments and have four dimensions.

It's three layers include

a) Output-Dim (Output layer: number of nodes in output layer will be same as input layer).
b) Hidden-Dim (Hidden layer: size of hidden layer. It the size of hidden-start of the LSTM).
c) Input-Dim (Total number of unique word in sample data).
d) Embedding-dim (Size of each embedding vector. Here embedding dimension of GloVe word embedding vectors is 100).

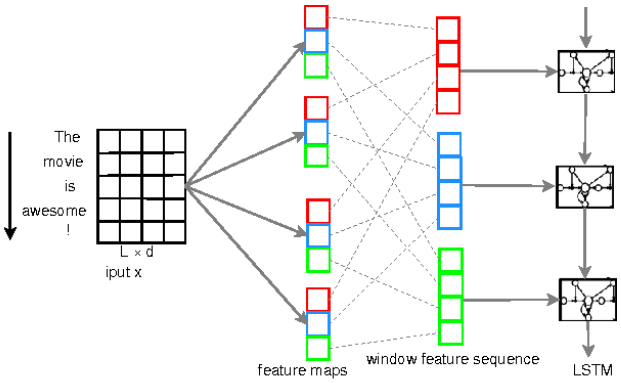

Figure 2 LSTM Model

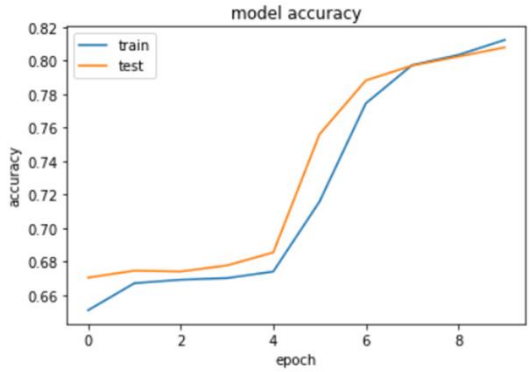

Figure 3 Model Accuracy Graph

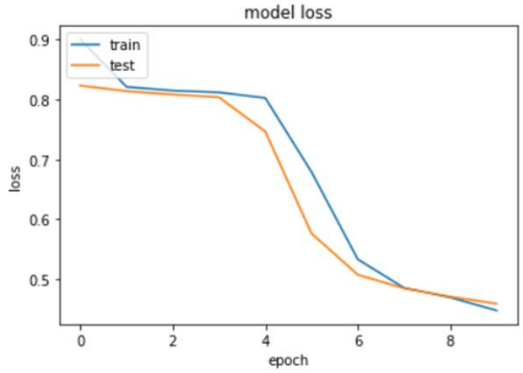

Figure 4 Model Loss Graph

5. Conclusion

This thesis aims at automated classification of bugs in gaming reviews to facilitate developers about what they can improve in games. Existing literature have very serious limitations in terms of multi label classification of bugs in gaming industry. Most of the existing literature is limited to abstract level of bugs classifications. In addition, automated model for the classification of multi-label gaming reviews is rarely seen.

State of the art studies have given the taxonomies and studies for classifications of bug which require an immediate need of more comprehensive way to detect it from user reviews. In this research we have developed a LSTM algorithm which classify the bugs into specified labels.

For automated multi-label bugs classification of gaming reviews this research implies machine learning based models and deep learning based models such as CNN, RNN and LSTM. These models are implemented using python and results are reported in the pilot study chapter. On the basis of results accuracies of pilot studies an automated multi-label bug classification model LSTM is implemented using python with fifteen different games with 10,000 reviews dataset. To classify the bugs into multi label classification different objectives were defined and proved on the course of this research.

6. Limitations and Future Work

In the context of ongoing work, the proposed multi-label bug classification model focuses mainly on classifying the bugs into fifteen categories which are defined in existing literature [7]. Also the number of epochs are set to minimum which can also be increased to achieve the better results. This implementation has used the GloVe word embedding technique however other embedding techniques like word2Vec can also be used. The proposed model is evaluated on the basis of accuracy other factors can also be considered to measure its accuracy.

REFERENCES

[1] D. Lin, C. P. Bezemer, Y. Zou, and A. E. Hassan, *An empirical study of game reviews on the Steam platform*, vol. 24, no. 1. Empirical Software Engineering, 2019.

[2] S. Iftikhar, M. Z. Iqbal, M. U. Khan, and W. Mahmood, "An automated model based testing approach for platform games," in *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 2015, pp. 426–435, doi: 10.1109/MODELS.2015.7338274.

[3] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *2013 21st IEEE International Requirements Engineering Conference (RE)*, 2013, pp. 125–134, doi: 10.1109/RE.2013.6636712.

[4] D. Lin, C. P. Bezemer, and A. E. Hassan, "Identifying gameplay videos that exhibit bugs in computer games," *Empir. Softw. Eng.*, vol. 24, no. 6, pp. 4006–4033, 2019, doi: 10.1007/s10664-019-09733-6.

[5] C. C. Aggarwal and C. X. Zhai, *Mining text data*, vol. 9781461432. 2013.

[6] A. Truelove, E. S. de Almeida, and I. Ahmed, "We'll Fix it in Post: What Do Bug Fixes in Video Game Update Notes Tell Us?," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2021, pp. 258–259, doi: 10.1109/ICSE-Companion52605.2021.00120.

[7] C. Lewis, J. Whitehead, and N. Wardrip-Fruin, "What went wrong: A taxonomy of video game bugs," *FDG 2010 - Proc. 5th Int. Conf. Found. Digit. Games*, pp. 108–115, 2010, doi: 10.1145/1822348.1822363.

[8] M. Westerdahl, V. Bahtijar, and D. Spikol, "Challenges in video game development," 2019.

[9] J. Iqbal *et al.*, *Requirements engineering issues causing software development outsourcing failure*, vol. 15, no. 4. 2020.

[10] Institute of Electrical and Electronics Engineers, “IEEE Standard Glossary of Software Engineering Terminology,” *Office*, vol. 121990, no. 1, p. 1, 1990, [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=159342.

[11] M. Klassen, S. Denman, and H. Driessen, “Requirements quality for a virtual world,” *Proc. - 15th IEEE Int. Requir. Eng. Conf. RE 2007*, pp. 375–376, 2007, doi: 10.1109/RE.2007.53.

[12] A. Hussain, O. Asadi, and D. J. Richardson, “A holistic look at requirements engineering practices in the gaming industry,” no. Section 4, 2018, [Online]. Available: http://arxiv.org/abs/1811.03482.

[13] J. P. Zagal, A. Ladd, and T. Johnson, “Characterizing and understanding game reviews,” *FDG 2009 - 4th Int. Conf. Found. Digit. Games, Proc.*, no. McCrea 2007, pp. 215–222, 2009, doi: 10.1145/1536513.1536553.

[14] W. Maalej, M. Nayebi, and G. Ruhe, “Data-Driven Requirements Engineering-An Update,” *Proc. - 2019 IEEE/ACM 41st Int. Conf. Softw. Eng. Softw. Eng. Pract. ICSE-SEIP 2019*, pp. 289–290, 2019, doi: 10.1109/ICSE-SEIP.2019.00041.

[15] L. F. S. Britto and L. D. S. Pacífico, “Evaluating Video Game Acceptance in Game Reviews using Sentiment Analysis Techniques,” *XIX Brazilian Symp. Comput. Games Digit. Entertain.*, pp. 399–402, 2020.

[16] H.-N. Kang, “A Study of Analyzing on Online Game Reviews Using a Data Mining Approach: STEAM Community Data,” *Int. J. Innov. Manag. Technol.*, vol. 8, no. 2, pp. 90–94, 2017, doi: 10.18178/ijimt.2017.8.2.709.

[17] J. Bergdahl, C. Gordillo, K. Tollmar, and L. Gisslén, “Augmenting Automated Game Testing with Deep Reinforcement Learning,” in *2020 IEEE Conference on Games (CoG)*, 2020, pp. 600–603, doi: 10.1109/CoG47356.2020.9231552.

[18] A. Di Sorbo, S. Panichella, C. V. Alexandru, C. A. Visaggio, and G. Canfora, “SURF: Summarizer of user reviews feedback,” *Proc. - 2017 IEEE/ACM 39th Int. Conf. Softw. Eng. Companion, ICSE-C 2017*, no. i, pp. 55–58, 2017, doi: 10.1109/ICSE-C.2017.5.

[19] A. Di Sorbo *et al.*, “What would users change in my App? Summarizing app reviews for recommending software changes,” *Proc. ACM SIGSOFT Symp. Found. Softw. Eng.*, vol. 13-18-Nove, pp. 499–510, 2016, doi: 10.1145/2950290.2950299.

[20] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, “AR-miner: Mining informative reviews for developers from mobile app marketplace,” *Proc. - Int. Conf. Softw. Eng.*, no. 1, pp. 767–778, 2014, doi: 10.1145/2568225.2568263.

[21] D. Lin, C. P. Bezemer, Y. Zou, and A. E. Hassan, *An empirical study of game reviews on the Steam platform*, vol. 24, no. 1. Empirical Software Engineering, 2019.

[22] S. Iftikhar, M. Z. Iqbal, M. U. Khan, and W. Mahmood, “An automated model based testing approach for platform games,” in *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 2015, pp. 426–435, doi: 10.1109/MODELS.2015.7338274.

[23] D. Pagano and W. Maalej, “User feedback in the appstore: An empirical study,” in *2013 21st IEEE International Requirements Engineering Conference (RE)*, 2013, pp. 125–134, doi: 10.1109/RE.2013.6636712.

[24] D. Lin, C. P. Bezemer, and A. E. Hassan, “Identifying gameplay videos that exhibit bugs in computer games,” *Empir. Softw. Eng.*, vol. 24, no. 6, pp. 4006–4033, 2019, doi: 10.1007/s10664-019-09733-6.

[25] C. C. Aggarwal and C. X. Zhai, *Mining text data*, vol. 9781461432. 2013.

[26] A. Truelove, E. S. de Almeida, and I. Ahmed, “We’ll Fix it in Post: What Do Bug Fixes in Video Game Update Notes Tell Us?,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2021, pp. 258–259, doi: 10.1109/ICSE-Companion52605.2021.00120.

[7] C. Lewis, J. Whitehead, and N. Wardrip-Fruin, “What went wrong: A taxonomy of video game bugs,” *FDG 2010 - Proc. 5th Int. Conf. Found. Digit. Games*, pp. 108–115, 2010, doi: 10.1145/1822348.1822363.

[8] M. Westerdahl, V. Bahtijar, and D. Spikol, “Challenges in video game development,” 2019.

[9] J. Iqbal *et al.*, *Requirements engineering issues causing software development outsourcing failure*, vol. 15, no. 4. 2020.

[10] Institute of Electrical and Electronics Engineers, “IEEE Standard Glossary of Software Engineering Terminology,” *Office*, vol. 121990, no. 1, p. 1, 1990, [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=159342.

[27] M. Klassen, S. Denman, and H. Driessen, “Requirements quality for a virtual world,” *Proc. - 15th IEEE Int. Requir. Eng. Conf. RE 2007*, pp. 375–376, 2007, doi: 10.1109/RE.2007.53.

[28] A. Hussain, O. Asadi, and D. J. Richardson, “A holistic look at requirements engineering practices in the gaming industry,” no. Section 4, 2018, [Online]. Available: http://arxiv.org/abs/1811.03482.

[29] J. P. Zagal, A. Ladd, and T. Johnson, “Characterizing and understanding game reviews,” *FDG 2009 - 4th Int. Conf. Found. Digit. Games, Proc.*, no. McCrea 2007, pp. 215–222, 2009, doi: 10.1145/1536513.1536553.

[30] W. Maalej, M. Nayebi, and G. Ruhe, “Data-Driven Requirements Engineering-An Update,” *Proc. - 2019 IEEE/ACM 41st Int. Conf. Softw. Eng. Softw. Eng. Pract. ICSE-SEIP 2019*, pp. 289–290, 2019, doi: 10.1109/ICSE-SEIP.2019.00041.

[31] L. F. S. Britto and L. D. S. Pacífico, “Evaluating Video Game Acceptance in Game Reviews using Sentiment Analysis Techniques,” *XIX Brazilian Symp. Comput. Games Digit. Entertain.*, pp. 399–402, 2020.

[32] H.-N. Kang, “A Study of Analyzing on Online Game Reviews Using a Data Mining Approach: STEAM Community Data,” *Int. J. Innov. Manag. Technol.*, vol. 8, no. 2, pp. 90–94, 2017, doi: 10.18178/ijimt.2017.8.2.709.

[33] J. Bergdahl, C. Gordillo, K. Tollmar, and L. Gisslén, “Augmenting Automated Game Testing with Deep Reinforcement Learning,” in *2020 IEEE Conference on Games (CoG)*, 2020, pp. 600–603, doi: 10.1109/CoG47356.2020.9231552.