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

Submitted by

ALI SHAHBAZ
(20I-2020)

Supervised by

DR. KHUBAIB AMJAD ALAM

Masters of Science (Software Engineering)

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science (Software Engineering) at National University of Computer & Emerging Sciences

Department of Computer Science

National University of Computer and Emerging Sciences

Islamabad, Pakistan

June 2022

Plagiarism Undertaking

I take full responsibility of the research work conducted during the Master's Thesis titled Automated Bugs Identification Through Early Access Game Review Analytics On Game Distribution Platforms. I solemnly declare that the research work presented in the thesis is done solely by me with no significant help from any other person; however, small help wherever taken is duly acknowledged. I have also written the complete thesis by myself. Moreover, I have not presented this thesis (or substantially similar research work) or any part of the thesis previously to any other degree awarding institution within Pakistan or abroad.

I understand that the management of National University of Computer and Emerging Sciences has a zero tolerance policy towards plagiarism. Therefore, I as an author of the above-mentioned thesis, solemnly declare that no portion of my thesis has been plagiarized and any material used in the thesis from other sources is properly referenced. Furthermore, the work presented in the thesis is my own original work and I have positively cited the related work of the other researchers by clearly differentiating my work from their relevant work.

I further understand that if I am found guilty of any form of plagiarism in my thesis work even after my graduation, the University reserves the right to revoke my Master's degree. Moreover, the University will also have the right to publish my name on its website that keeps a record of the students who plagiarized in their thesis work.

_____

ALI SHAHBAZ

Date: _____

Author's Declaration

I, Ali Shahbaz, hereby state that my Master's thesis titled Automated Bugs Identification Through Early Access Game Review Analytics On Game Distribution Platforms my own work and it has not been previously submitted by me for taking partial or full credit for the award of any degree at this University or anywhere else in the world. If my statement is found to be incorrect, at any time even after my graduation, the University has the right to revoke my Master's degree.

_____

ALI SHAHBAZ

Date: _____

Certificate of Approval

It is certified that the research work presented in this thesis, entitled "Automated Bugs Identification Through Early Access Game Review Analytics On Game Distribution Platforms" was conducted by Ali Shahbaz under the supervision of Dr. Khubaib Amjad Alam.

No part of this thesis has been submitted anywhere else for any other degree.

This thesis is submitted to the Department of Computer Science in partial fulfillment of therequirements for the degree of Masters of Science in Computer Science

at the

National University of Computer and Emerging Sciences, Islamabad, Pakistan

June 2022

Candidate Name: Ali Shahbaz Signature: _____

Examination Committee:

1. Name: Dr. Irum Inayat Signature: _____
   Assistant Professor, FAST-NU Islamabad.
2. Name: Dr. Uzma Bibi Signature: _____
   Assistant Professor, FAST-NU Islamabad

Dr. Hammad Majeed _____
Graduate Program Coordinator, National University of Computer and Emerging Sciences, Islamabad, Pakistan.

Dr. Waseem Shahzad _____
Head of the Department of Computer Science, National University of Computer and Emerging Sciences, Islamabad, Pakistan.

Abstract

User reviews are considered one of the most important source of information about an app and game. The classification and analysis of reviews in order to extract information has proven to be a considerable difficulty. 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. Because of the enormous user base and potential benefits of automated feature and bug extraction, game application review analysis has lately arisen as an active topic of research in software engineering. 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.

Acknowledgement

First and foremost, I have to thank Almighty Allah for countless blessings. ThenI have to thank my parents for their love and support throughout my life. Thankyou both for giving me strength to reach for the stars and chase my dreams. My brothers, and my entire family deserve my wholehearted thanks as well. I wouldlike to sincerely thank my supervisor, Dr. Khubaib Amjad Alam, for his guidanceand support throughout this research, and especially for his confidence in me. I would also like to thank all friends who stood by me through the ups and downs of life.

Table of Contents

List of Figures

Chapter 1

1. Introduction

Computer games have grown in popularity throughout time, resulting in a multi-billion-dollar industry. The size of the computer gaming industry makes it difficult to create a successful game. In addition, prior studies show that gamers are extremely hard to please, making the quality of games an important issue. 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]

Application distribution platforms, such as Google Play or Apple App Store, allow users to rate and review downloaded applications. The popularity of these platforms among both application developers and users has been on the rise in recent years. However, their potential to and impact on requirements engineering processes are not yet well understood.[3] With the growing market and competition in the gaming industry, it is important to produce high-quality games in order to succeed. Developers commonly use bug reports from gamers to locate bugs in their games. Recently, gameplay videos have become popular in the gaming community Some of these videos showcase a bug, providing developers with new opportunities to collect rich bug information. [4]

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. 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]

If a bug persists into a game's release, it can have a negative impact on both the developers and users. Despite this Effects, it is common to launch games with existing bugs Fixed by subsequent updates in 2014, for example, three-quarters of big-budget games "released on Xbox One, Wii U and PS4" received an update within 24 hours of the game's initial release.

The different development processes for games and traditional software may be responsible for the appearance of these bugs. Previous research has shown that it is difficult to test all aspects of a video game comprehensively. 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 for some.[7]. This challenge draws attention to a more comprehensive analysis of re-views to identify issues and requirements, given the potential value that the text of the review means. As per the above discussion there was a need of tool which takes a user review dataset and then identify bugs form 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.

1.1. Challenges Related to Game Reviews Analysis

Despite the fact that the video game industry has changed dramatically over the previous few decades, multiple reports reveal that creators still confront numerous problems in their everyday job. One of the most difficult challenges is getting projects to finish on schedule and within budget.[8] Efficient requirement gathering not only speeds up the development process, but also results in more efficient and cost-effective solutions.[9] IEEE defines requirement as “a condition or capability needed by a user to solve a problem or achieve an objective”. [10] The challenges are deeper in the case these products are mobile video games.
In particular, mobile games need to meet many important non-functional requirements (portability, gameplay, emotional aspects, etc.). In addition, because mobile games are developed for the mass market, development teams need to understand the needs of very different and sometimes unfamiliar stakeholders. [11] Requirements engineering practices in the gaming industry differ from those in the traditional software industry. [12]
As per the above limitations developers need to identify reported issues and suggestions by analyzing and categorizing a large corpus of reviews. Game reviews are rich and diverse in terms of topics and topics covered, and when viewed purely as a "buyer's guide", games, game developers, it provides readers with advice on how to approach a particular game and enjoy it to the fullest. More generally, game reviews also help keep a history of video games by contextualizing the connections and history connections that exist between games.[13] While these categories are valuable for improving the capabilities of your game, they need bug detection and classification when it comes to quality and user satisfaction.

1.2. Problem Formulation

Mining and analyzing user game reviews has emerged as an important challenge for game developers. There are millions of games in the highly competitive gaming market, and these games have billions of user reviews. This excessive amount of user ratings makes it difficult to analyze and manually interpret the ratings for information extraction.
Software vendors are accumulating huge amounts. Implicit feedback in the form of usage data, error logs, and sensor data. These trends point to a shift to data-driven user-centered identification, prioritization, and management of software requirements. While dealing with large set of data training of classification models are performed using feature extraction. [14]
However, 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. The previous studies have more complex methods to extract features from the reviews. [15], [16]. However, 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.

1.2.1. Game reviews Classification

The classification model for analyzing Game reviews automatically assigns reviews to classes while monitoring the content of the review text [17]. To train and evaluate these models, you need data from labeled evaluations. In this data, ratings are manually categorized by humans into various categories. In addition to the quality of the labeled data used to train the model, the accuracy of the classification depends on the set of features identified from the review text.
This thesis evaluates performance of deep learning models with different classification of features. Different techniques along with accuracies are presented in the model implementation chapter.

1.2.2. Multi-Label Game Reviews Analysis

Previous studies have classified Game reviews as one or more classes. [18], [19], [20], [21] Most users point out multiple bugs/Issues in a single rating and identify and categorize multiple categories in a single rating, but confirming a complete rating of a review is a multi-labeled classification problem. To address this challenge, we performed a multi-label classification using a problem transformation approach on a deep learning-based model.
In addition to the review's multi-label classification problem, feature engineering for review classification can introduce fake feature extraction tasks when considering irrelevant information present in user reviews. To address this issue, a new domain of ML

models (called deep learning) has recently been introduced that automatically learns features from tagged data. [19]
CNN (convolutional neural network) a deep learning model perform well on text classifications problems. But it's performance is unknown for multi label text classification problems that's why we are performing deep learning analysis of multiple models including LSTM, RNN (long term short term memory, Recurrent neural network). To compare their accuracy in the end. And using the one which evaluate results with high accuracy. In this thesis first, we labeled the dataset according to the bugs categories defined in the literature [7]. Categorize them into multiple categories of bugs and trained multiple deep learning models to get information from the models.

1.3. Research Objectives

This research mainly focuses on the Multi-Label Review Classification problem in the context of video games. To resolve this problem, following objectives are intended to be achieved:

- RO1: To gain insights of the existing classification methods that need to be addressed in the context of game rating analysis.
- RO2: To evaluate the performance of deep learning models to multi-label text classification.
- RO3: To automatically categorize gaming reviews into multiple bugs labels using deep learning models (CNN, RNN & LSTM).

1.4. Research Questions

Following research questions have been identified according to the problem identified in the previous sections:

- RQ1: What is the most widely used classification approach for game reviews analysis?
- RQ2: How do feature extraction impact the performance of deep learning classification models for gaming reviews?
- RQ3: How reviews can be categorized into multiple labels using deep learning approaches CNN, RNN & LSTM?
- RQ4: What are gamers talking about in gaming reviews?

1.5. Research Approach

The whole research process is divided into four distinct phases. Below is the brief overview of the four phases of research process.

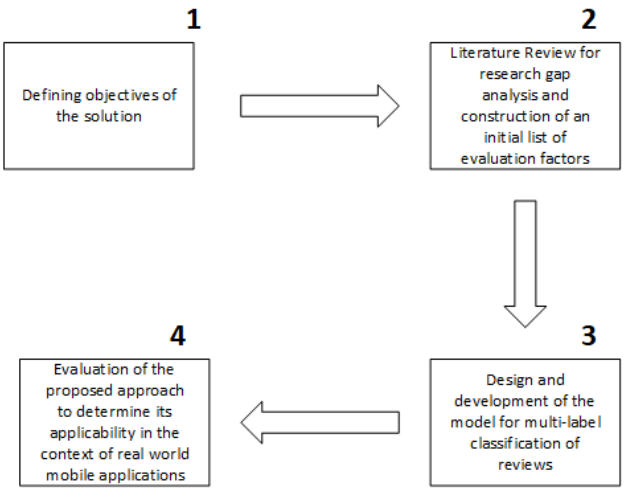


Figure 1.1 Research Methodology

First phase comprises of defining the objective of the solution which we are presenting in this problem. Second phase we are reviewing the literature for the identification of research gap and the information available related to automated classification of bugs from gaming reviews. Multiple perspectives are covered in this phase which include the multi-class text classification problem and multi label text classifications. In the third phase which is design and development of multi label classification model for gaming reviews analysis. Here multiple deep learning models are used to evaluate their performance on gaming reviews. And considered the one which have high accuracy for multi label reviews classification.
Later in the last phase model is evaluated by diving the dataset in eight twenty rule. And accuracy is calculated.
After successfully evaluating the model, the trained model is tested for real life unseen data.

1.6. Summary

This chapter provided an overview of this thesis and an overview and background of the issue in order to lay the groundwork for the initial investigation for the game reviews classification problem. This chapter contains specific problem areas, formalized research questions, and research objective achieved during the course of this research. The next chapter will comprehensively review the existing literature on Game reviews.

Chapter 2

2 Literature Review

This chapter first explain the background information of the problem then gaming reviews analysis of existing literature is discussed. Provide detail information of the automated machine learning models used in this thesis.

2.1 Automated Text Analysis

Every day, a large number of user reviews are published in text format on the gaming apps marketplace. Therefore, the automated approach facilitates the manual task of extracting critical information from user reviews and supports a variety of game development release cycle activities. Example of a user game review, containing useful information for the improvement of a game is shown in figure: Gaming review.
Traditional automated approached used to mine the reviews are based on supervised machine learning techniques. [22][23][24]

Figure 2.1 Game Review

In Supervised machine learning model, the algorithm learns from the input output training pair and maps the sample according to the learned mapping function. The training set considered as an example is R = (Xi, Yi)^N. Where N is the number of reviews. The review is considered full text and the target label is manually assigned to the review. Where Yi indicates the category or label that classifies the reviews into bug categories. For classification model dataset is transformed into vector format of zero and one such that number one is assigned to a review which belong to the labeled class and others are set to zero. Each review is represented as a vector x = (xi(1) , xi(2) , ..., xi(m)) length m. Below figure is the example of how one review can contain multiple labels.

Reviews	Labels
I returned it 3d interface impossible to negotiate even looking at the side view then the complete the tutorial keep an eye on the clock. Score was not updating.	Graphical error, Invalid value
still requires you to play online on top of it I suddenly have massive technical issues with the connection.	Invalid context over time

Graphical Error	Invalid Value Change	Invalid Context overtime
1	1	0
0	0	1

Figure 3 Reviews Encoding Example

The training example helps you learn the mapping function used to predict hidden data that will not be used during training. For example, the output of the results from the training model predicts a defined set of possible labels.

2.1.1 Machine Learning for Reviews Classification

ML is primarily based on the concept of pattern-based decision-making with less human intervention, creating models that automate analysis, training the models, learning different patterns from data, and Artificial Intelligence (AI). Researchers are studying AI to see how machines learn from data by performing specific tasks according to learning patterns, without the explicit programming that forms the basis of ML. What is important is the repeatability of the ML. This allows the model to independently adapt new data and learn from previously trained models to produce reproducible and reliable results.
Studies also revealed that machine learning approaches are best for text classification and analysis. [22][23][24]. Another approach include deep learning is also the part of the machine learning approaches. Below diagram represent the machine learning approaches

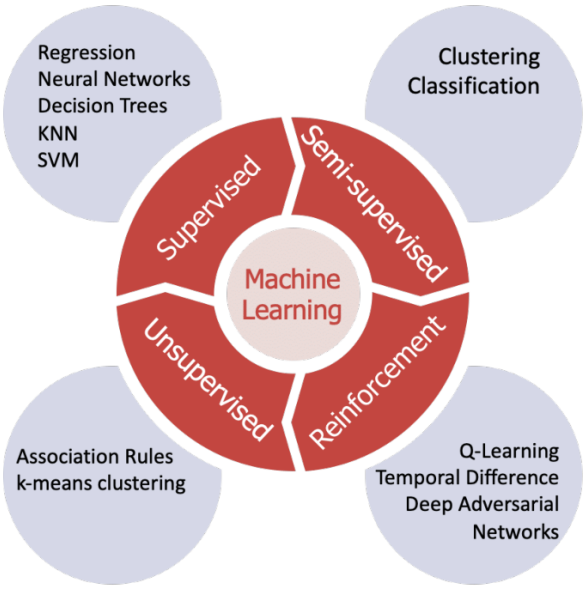


Figure 4 Machine Learning Approaches

Unsupervised Learning:

The category of ML, in which the model is well trained for observation without prior output, can allow the autonomous construction of new models resulting from the data. [25] This type of machine learning algorithm finds hidden patterns in unlabeled input data, as opposed to supervised machine learning methods that use labeled data to train the model. The most commonly used algorithm in unsupervised machine learning is K-mean clustering. [26]
Simple unsupervised learning algorithm for classifying reviews as recommended (thumbs up) or not recommended (thumbs down) by D. Turney. [27]

Reinforcement Learning:

Reinforcement learning considers learning from the environment and taking action based on the experience learned. Researchers have used this technique in many ways such as Zhang provided an RL method for learning sentence patterns by discovering the structure of related tasks. [28]

Supervised Learning:

Supervised machined learning dataset is labeled and output is known. It is most widely used branch in machine learning. Many reviews analysis approached have implemented using supervised machine learning. [29][18][30].

Supervised Machine Learning Algorithms:

Below is the overview of supervised machine learning algorithms which are used in this thesis for reviews text classification.

Deep Learning:

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. They are comprised of several processing layers. They have minimized the effort of manual feature engineering which is used in traditional machine learning models. It has been used for text classification in different research domains. Recently various deep learning models are used for text classification and analysis but accuracy of these models for this problem is unknown [31][32][33][34][35][36][37]. Therefore, we are using RNN, CNN and LSTM deep learning models which is evaluated for game reviews bug classification. Below section describe gives the overview of the working of these models.

Recurrent Neural Network:

A recurrent neural network (RNN) is able to process a sequence of arbitrary length by recursively applying a transition function to its internal hidden state vector ht of
the input sequence. The activation of the hidden state ht at time-step t is computed as a function f of the current input symbol xt and the previous hidden state ht1.

$$\mathbf{h}_t = \begin{cases} 0 & t = 0 \\ f(\mathbf{h}_{t-1}, \mathbf{x}_t) & \text{otherwise} \end{cases} \qquad (1)$$

It is common to use the state-to-state transition function f as the composition of an element-wise nonlinearity with an affine transformation of both xt and ht1. Traditionally, a simple strategy for modeling sequence is to map the input sequence to a fixed-sized vector using one RNN, and then to feed the vector to a soft max layer for classification or other tasks [38]
Let's define the equations needed for training:

1) $h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t)$
2) $y_t = softmax(W^{(S)}h_t)$
3) $J^{(t)}(\theta) = \sum_{i=1}^{|V|} (y_{t_i}^{'} log y_{t_i})$

1) —holds information about the previous words in the sequence. As you can see, h_t is calculated using the previous h_(t-1) vector and current word vector x_t. We also apply a non-linear activation function f (usually tanh or sigmoid) to the final summation. It is

acceptable to assume that h_0 is a vector of zeros.
2) — calculates the predicted word vector at a given time step t. We use the softmax function to produce a (V,1) vector with all elements summing up to 1. This probability distribution gives us the index of the most likely next word from the vocabulary.
3) — uses the cross-entropy loss function at each time step t to calculate the error between the predicted and actual word. RNN working

Convolutional Neural Network:

The convolution layer uses filters that perform convolution operations as it is scanning the input I with respect to its dimensions. Its hyper parameters include filter size F and padding P. It uses Rectified linear unit (ReLU) for activation function. It aims at introducing non-linearity's to the network (g(z) = max (0, z)).
Filter hyper parameters: The convolution layer contains filters; in which following parameters are applied:

- Dimensions of a filter: Dimension is defined for an input, which will produce the feature map.
- Padding: Padding is applied, in which filter see the input end to end. It can be achieved by adding 0 to each side of the boundaries of the input.

The **pooling layer** is a down sampling operation, typically applied after a convolution layer, which does some spatial invariance. The **fully connected layer** operates on a flattened input where each input is connected to all neurons.
Activation function is applied to predict the output labels of the input sentences. [35]

Long Short-Term Memory:

Long Short-Term Memory (LSTM), a typical recurrent neural network which is widely used in text classification tasks. Recurrent neural networks, predictions are made sequentially (e.g., for the vector representation of each word), and the hidden layer from one prediction is fed to the hidden layer of the next prediction, making it capable of capturing the structure of the sentences. LSTM add additional factors to a traditional RNN that give it more of a fine-grained control over memory.
These factors control

1. How much the current input matters in creating the new memory?
2. How much the previous memories matters in creating the new memory?
3. What parts of the memory are important in generating the output?

Compared with traditional RNN, LSTM is good at handling long term dependency (i.e., a late output depends on a very early input) [39]
For example, {T1, …, Tn} are input to LSTM one by one in a sequence. each term Ti is first converted to the corresponding input vector xi using Word2Vec model and input into LSTM one by one. At each time j, the output W of the hidden layer Hj will be propagated back to the hidden layer together with the next input xj+1 at the next point of time j+1. Finally, the last output Wn will be fed to the output layer. [40]

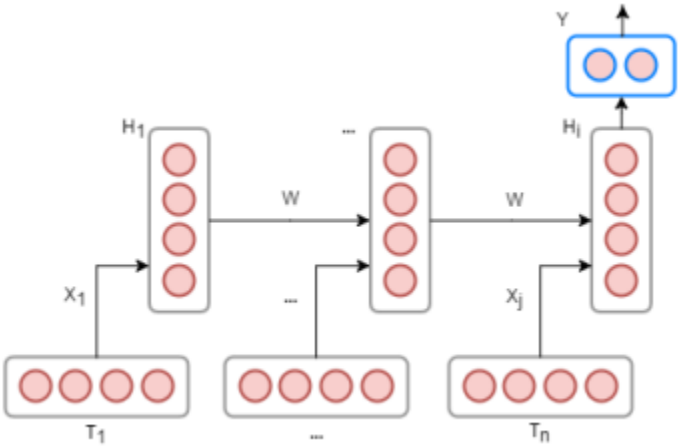


Figure 5 LSTM

2.2 Gaming Bugs Classification

We review the existing work related to automated classification of bugs from gaming reviews. Taxonomies and case studies are presented in the past to get information from gaming reviews which include user response toward the app, issues user are facing, the new improvements users want in game.

A study by [41] was presented name as PUMA, an auto-mated, phrase-based approach to extract user opinions in app reviews. This approach includes a technique to extract phrases in reviews using part-of-speech (POS) templates. Many gaming industries go through the gaming reviews manually also a research has been done in which they have examined more than 100 thousand bug reports of 380 projects, they found that bugs can be classified into four types based on the location of their fixes. They have divided the bugs into four different type. Type 1 bugs are the ones that fixed by modifying a single location in the code, while Type 2 refers to bugs that are fixed in more than one location. Type 3 refers to multiple bugs that are fixed in the exact same location. Type 4 is an extension of Type 3, where multiple bugs are resolved by modifying the same set of locations. [42] A probabilistic techniques are used to classify the reviews into four type: bug reports, feature requests, user experience and text ratings. This technique use review metadata such as the star rating and the tense, as well as, text classification, natural language processing, and sentiment analysis techniques. [43] Unity games are the most downloaded games in the word. In order to detect bad smells in the unity project a tool was developed call Unity Linter a static tool that support unity projects to detect bad smells from the projects. It had a precision of 80% to 100%. [44] Gaming bugs can also be detected in real time using deep enforcement learning technique [17]. Deep enforcement learning (DRL) can be used to increase test coverage, find exploits, test map difficulty, and to detect common problems that arise in the testing of first-person shooter (FPS) games. Gaming bugs had also been

detected from gameplay video using random forest classifier. Which rank gameplay videos based on their likelihood of being a bug video. This proposed approach achieves a precision that is 43% higher than that of the naive keyword searching approach on a manually labelled dataset of 96 videos. [4]

It is important factor that how much time user spend to review a game or app. To detect how long user, spend time to use the app or play the game an exploratory case study was presented. They Found that most of the feedback is provided shortly after new releases, with a quickly decreasing frequency over time.[3] To classify the bugs into specific categories based on their location a taxonomy was presented by [7]. A taxonomy of possible failures, divided into temporal and non-temporal failures. This taxonomy can guide the thinking of designers and testers alike, helping them expose bugs in the game. This taxonomy gives a pathway to automate the process of analyzing gaming reviews and divide them into related bugs categories.

Gaming review can contain more than one bug at the same time which can be multi-type of information. Thus user review can be labelled into more than one categories. [45] Various approaches exist to classify data having multiple labels including binary relevance, classifier chains, ensemble of classifier chains, and ensemble of binary relevance. Multi-label review classification problem, feature techniques for rating classification can lead to false feature extraction attempts, due to irrelevant information in the reviews. To solve this problem, an emerging field of machine learning model has been introduced in recent years, which automatically learns features from labeled data, known as Deep Learning. Which have proved efficient for various text classification problems [31][32][33][34][35][36][37]

2.3 Summary

This chapter presented the brief information about automated analysis of reviews using traditional and machine leaning methods. It describes the traditional and automated methods used in the analysis of reviews. Then the classification models that are used to solve this thesis problem are explained with its working steps.

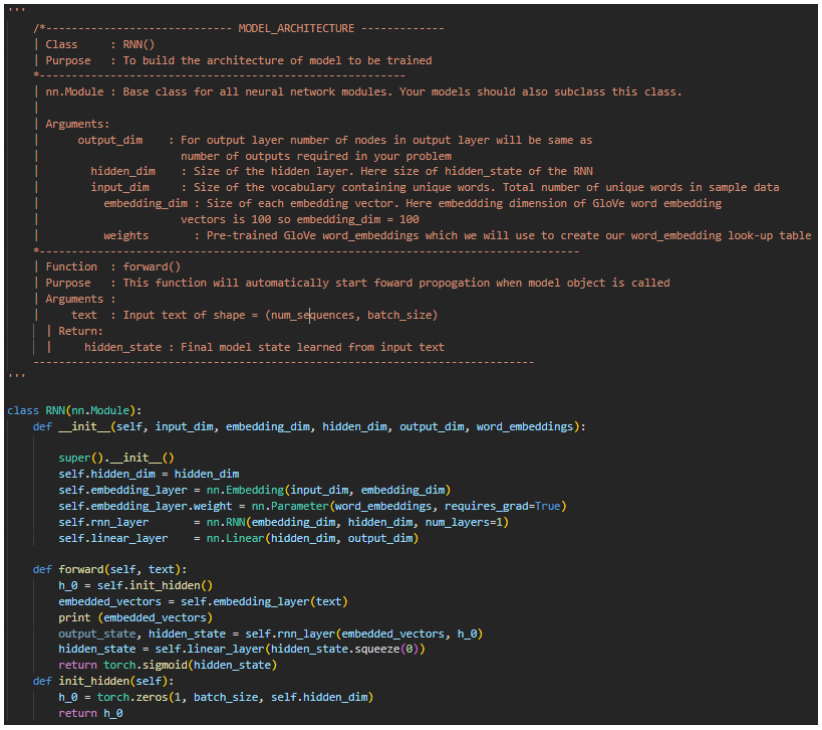
Chapter 3

3 Pilot Studies

Multiple deep learning approaches are available for multi label text classification. But which perform best for the multi label text classification was unknown. So in this section we conducted the pilot studies of multiple deep learning models on 1500 gaming reviews dataset. Which turn out much beneficial for the selection of right classification model. We implemented the most widely used text classification models RNN (Recurrent neural network), CNN (Convolutional neural network) and LSTM (Long Short-Term Memory). Results of these studies are discussed in this chapter. All the models are implemented using python 3 using JUPYTER notebook created in VS code.

3.1. RNN (Recurrent Neural Networks)

A Recurrent Neural Network (RNN) is a type of neural network in which the output of the previous step is provided as the input of the current step. RNN is mainly used for sequence classification - sentiment classification and video classification. Sequence Tagging - Marks part of speech and identifies a named entity. [38] First the data set was divided into **train set, validation set & test set**. GloVe embedding is used to implement word embedding. Before word embedding data is Pre-processed. Then the parameters are initialized which include input dimension, embedding dimension, hidden dimension, output dimensions and word embedding.



```
'''
    /*---------------------------- MODEL_ARCHITECTURE ------------
    | Class     : RNN()
    | Purpose   : To build the architecture of model to be trained
    *-------------------------------------------------------------
    | nn.Module : Base class for all neural network modules. Your models should also subclass this class.
    |
    | Arguments:
    |      output_dim    : For output layer number of nodes in output layer will be same as
    |                      number of outputs required in your problem
    |       hidden_dim    : Size of the hidden layer. Here size of hidden_state of the RNN
    |       input_dim     : Size of the vocabulary containing unique words. Total number of unique words in sample data
    |         embedding_dim : Size of each embedding vector. Here embeddding dimension of GloVe word embedding
    |                      vectors is 100 so embedding_dim = 100
    |          weights       : Pre-trained GloVe word_embeddings which we will use to create our word_embedding look-up table
    *-------------------------------------------------------------------------------
    | Function  : forward()
    | Purpose   : This function will automatically start foward propogation when model object is called
    | Arguments :
    |     text  : Input text of shape = (num_sequences, batch_size)
    | | Return:
    | |     hidden_state : Final model state learned from input text
    ---------------------------------------------------------------------------
'''

class RNN(nn.Module):
    def __init__(self, input_dim, embedding_dim, hidden_dim, output_dim, word_embeddings):

        super().__init__()
        self.hidden_dim = hidden_dim
        self.embedding_layer = nn.Embedding(input_dim, embedding_dim)
        self.embedding_layer.weight = nn.Parameter(word_embeddings, requires_grad=True)
        self.rnn_layer       = nn.RNN(embedding_dim, hidden_dim, num_layers=1)
        self.linear_layer    = nn.Linear(hidden_dim, output_dim)

    def forward(self, text):
        h_0 = self.init_hidden()
        embedded_vectors = self.embedding_layer(text)
        print (embedded_vectors)
        output_state, hidden_state = self.rnn_layer(embedded_vectors, h_0)
        hidden_state = self.linear_layer(hidden_state.squeeze(0))
        return torch.sigmoid(hidden_state)
    def init_hidden(self):
        h_0 = torch.zeros(1, batch_size, self.hidden_dim)
        return h_0
```

Figure 6 RNN Implementation

RNN is a feed forward model output of one node is the input of the next node. At first hidden start is set to all zeros. Here we have mapped all the indexes present tin the input sequence of corresponding word vector using our trained word embedding. Then Applied RNN layer to start learning of the words. Then at last linear layer applied to the output. Model accuracy is measured using the **Jaccard-Score and f1_score & hamming loss.** Model is later saved to test it for the real world input and calculated the accuracy which is shown in the below table.

Epochs	Accuracy	Val-Accuracy
1	0.6104	0.6000
2	0.6201	0.6198
3	0.6202	0.6101
Average	0.6169	0.6099

Table 1 RNN accuracy in 1500 reviews

3.2. LSTM (Long Short Term Memory)

LSTM stands for Long Short Term Memory. 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. Like other neural networks,

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 had widely been used for recent researches. [39][40] It's basic details are already described in chapter 3.
It is implemented using python torch (machine learning) library. Multiple layers of LSTM are implemented. Sigmoid function is used as an activation function. The details of this model is explained in the chapter 4.

```
'''
    /*---------------------------- MODEL_ARCHITECTURE ------------
    | Class     : LSTM()
    | Purpose   : To build the architecture of model to be trained
    *-------------------------------------------------------------
    | nn.Module : Base class for all neural network modules. Your models should also subclass this class.
    |
    | Arguments:
    |      output_dim    : For output layer number of nodes in output layer will be same as
    |                      number of outputs required in your problem
    |       hidden_dim   : Size of the hidden layer. Here size of hidden_state of the LSTM
    |       input_dim    : Size of the vocabulary containing unique words. Total number of unique words in sample data
    |         embedding_dim : Size of each embedding vector. Here embeddding dimension of GloVe word embedding
    |                      vectors is 100 so embedding_dim = 100
    |         weights      : Pre-trained GloVe word_embeddings which we will use to create our word_embedding look-up table
    *-------------------------------------------------------------------------------------
    | Function  : forward()
    | Purpose   : This function will automatically start foward propogation when model object is called
    | Arguments :
    |     text  : Input text of shape = (num_sequences, batch_size)
      | Return:
      |     hidden_state : Final model state learned from input text
    ---------------------------------------------------------------------------
'''

class LSTM(nn.Module):
    def __init__(self, input_dim, embedding_dim, hidden_dim, output_dim, word_embeddings):

        super().__init__()

        self.hidden_dim = hidden_dim
        self.embedding_layer = nn.Embedding(input_dim, embedding_dim)         # Embedding layer shape
        # Assign pre-trained weights and train during model training
        # So the weight will be updated during backpropogation(if you don't want to train them during model training set requires_grad = False))
        self.embedding_layer.weight = nn.Parameter(word_embeddings, requires_grad=True)
        self.lstm_layer       = nn.LSTM(embedding_dim, hidden_dim, num_layers=1) # We can implement multiple layers of LSTM simply by changing num_layers value
        self.linear_layer     = nn.Linear(hidden_dim, output_dim)             # Shape of linear layer

    def forward(self, text):

        h_0, c_0 = self.init_hidden()   # Initialize first hidden state to all zeros

        # Here we will map all the indexes present in the input sequence to the corresponding
            # word vector using our trained word_embedddings.
          # embedded input of shape = (num_sequences, batch_size, embedding_dimension)
        embedded_vectors = self.embedding_layer(text)
        print (embedded_vectors)
        output_state, (hidden_state, cell_state) = self.lstm_layer(embedded_vectors, (h_0, c_0))  # Apply lstm layer and start learning sequence of words
        hidden_state = self.linear_layer(hidden_state.squeeze(0))      # Apply the linear layer on output
        return torch.sigmoid(hidden_state)
    def init_hidden(self):
        h_0 = torch.zeros(1, 1, self.hidden_dim)
        c_0 = torch.zeros(1, 1, self.hidden_dim)
        return h_0, c_0
```

Figure 7 LSTM Implementation

The results for the model accuracy for 1500 reviews for 3 epochs are given in the table below:

Epochs	Accuracy	Val-Accuracy
1	0.6404	0.6400
2	0.6401	0.6498
3	0.6402	0.6401
Average	0.6402	0.6433

Table 2 LSTM model accuracy on 1500 reviews

3.3. CNN (Convolutional Neural Network)

It is implemented using python torch (machine learning) library. Three layers of model are used which include pooling layer, convolutional layer and last one is the fully connected layer. In the first layer which is pooling layer dimensions of the feature map are reduced. It reduced the number of parameters to learn and amount of computation in the network. In the convolutional layer filter map is applied on original dataset to feature maps in CNN model. There is where most of the user specified parameters are in the network. Here kernel size is also defined. And the fully connected layer contained the final model output which have connected nodes with the output labels.
CNN was also modified for the multi-label text classification because mostly it using for multi label image classification problems. We customized it multi label bug classification model which specifies that:

- Number of label matches the number of nodes on output layer.
- Activation function Sigmoid is used for each node in the output layer.
- Binary Cross-entropy loss function to compare each of the predicated probabilities to actual class output which can be 0 or 1.

The results of the 3 epochs for the testing data set of 1500 reviews are presented in the below table:

Epochs	Accuracy	Val-Accuracy
1	0.4904	0.4900
2	0.5001	0.4993
3	0.5002	0.5000
Average	0.4969	0.4963

Table 3 CNN model accuracy on 1500 reviews

3.4. Summary

This chapter gives an overview of different deep learning algorithms accuracy on the same dataset. This study had helped us to continue with the best fit model for the multi-label game reviews classification which have higher accuracy. Furthermore, it shows the accuracies of other deep learning models. And the implementation details are discussed.

Chapter 4

4 Proposed Classification Model

This chapter highlight the problem which is discussed in chapter 1 and its proposed solutions. First section describes the high level architecture of the proposed classification model. Second section describes the concepts and plans needed for designing the classification model. And in the last section multiple phases of the proposed solution is

presented.

4.1. Architecture of the Proposed Model

As per the chapter 3 LSTM model have shown higher accuracy for Multi-label text classification. In this section high-level architecture for the implementation of the proposed classification model LSTM is shown. This architecture mainly consists of reviews scrapping (collection of reviews), text preprocessing then classification model shown in figure 6. In the first phase reviews are scrapped from **STEAM ENGINE GAME PLATFORM** using customized web scrapper known as **SCRAPY**. To save time and effort this tool is designed in a way that it can scrap reviews of the game by inputting the games ID's (every game has unique ID on game steam platform). The reviews are in JSON format are converted into CSV format for the Pre-Processing techniques. Rather than dealing with the reviews one by once whole reviews are combined by removing the special character (,) and (.) from the reviews.

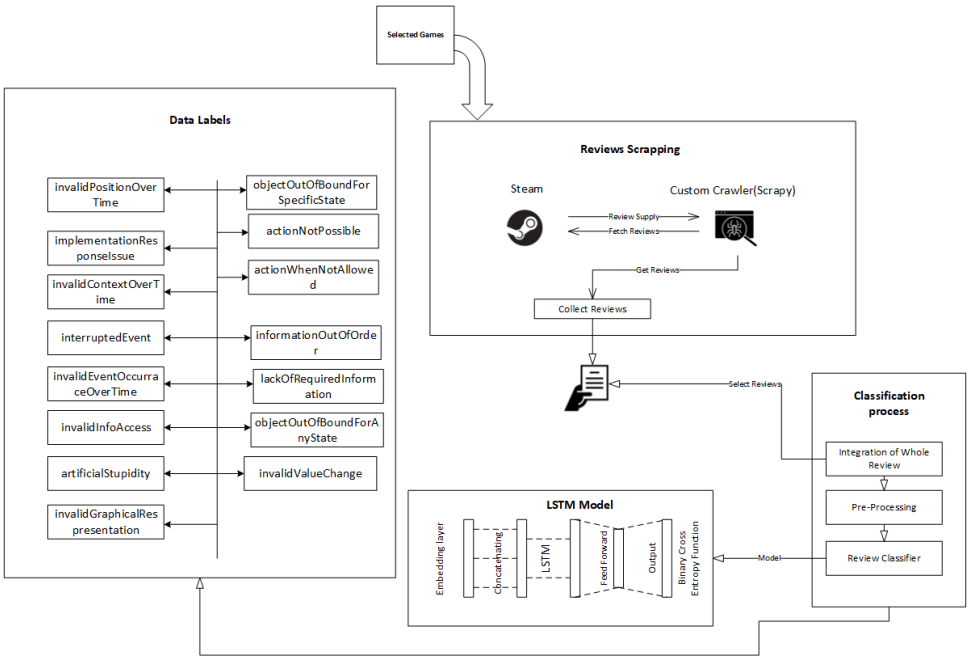


Figure 8 Architecture Diagram of Proposed Solution

Pre-processing was required because data is in textual format. And textual format contains so much other information which is not related to our problem and its solution. Reviews dataset contain many special symbols and other words which can impact the model performance and reduce its accuracy. So in order to increase the performance of the model these symbols and punctuations were removed using Natural Language Processing (NLP) techniques Tokenization, Stop Words Removal, Stemming & Lemmatization.
After the Pre-processing multi-label text classification Long Short Term Memory (LSTM)

is implemented into defined set of labels. As we cannot give input to deep learning model in textual format so it need to be first converted into vector format. For the word embedding we have used **GloVe word embedding vectors 100d** (available in kaggle dataset).

4.2. Deep Learning Concepts

4.2.1. Word Embedding

Textual data can be directly inputted to deep learning model. It need to be converted into vector format. Then the model is initialized. There are multiple embedding techniques which are available e.g. word2vec, GloVe. We are using GloVe embedding to encode the input data into vectors.

4.2.2. Long Short Term Memory

Binary Cross Entropy Function

Binary cross entropy compares each predicted probability to the actual class output (0 or 1). Then calculate the score that penalizes the probability based on the distance from the expected value. That means how close or far it is to the actual value.

Classification Process

Classification steps is comprising of four steps:

- User reviews collection
- Pre-Processing
- LSTM classification model
- Validation

Figure 7 show the generic approach of the classification model.

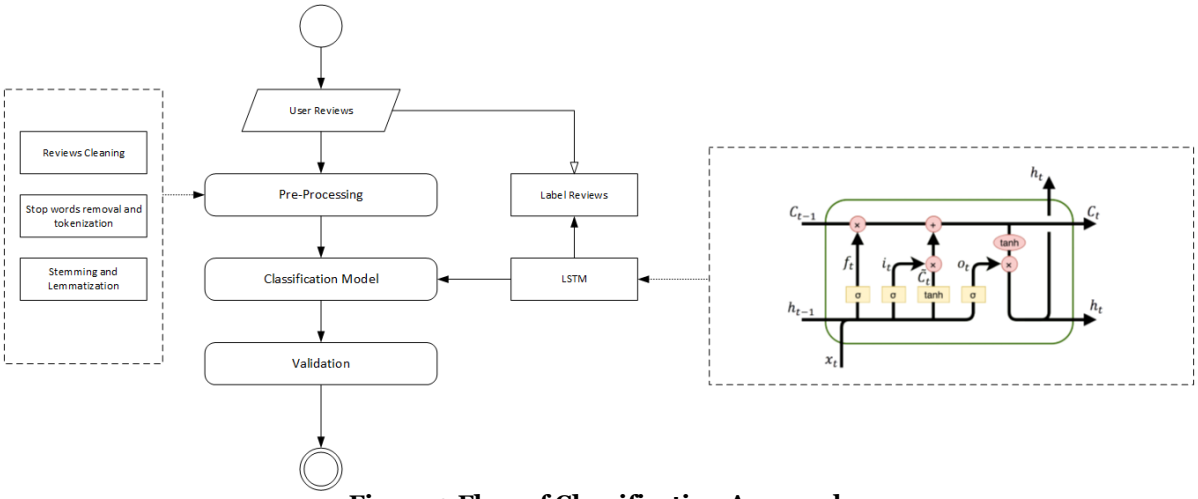


Figure 9 Flow of Classification Approach

4.3. LSTM Design

The proposed LSTM model working is shown in figure 8. LSTM receives five arguments and have four dimensions.
It's four layers include

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

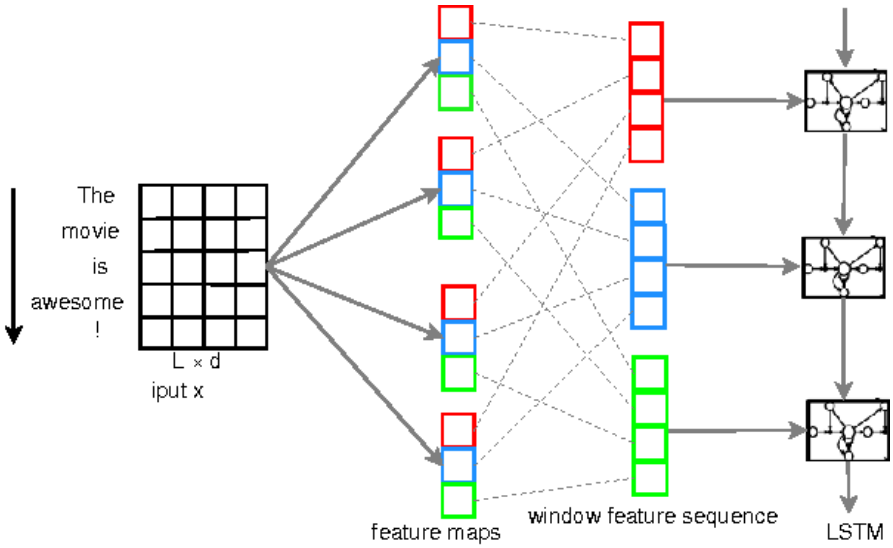


Figure 10 LSTM Model

4.4. Evaluation

Accuracy is a simple estimation statistic and calculates the total proportion of correctly classified samples. Accuracy is computed as a percentage of the numbers of correctly categorized occurrences under a particular label (TP) a total number of classified occurrences under the identical label (TP + FP). We have applied Test Hamming and Test F1 also to calculate the accuracy of the results.

4.5. Summary

This chapter summarized the proposed solution of the problem discussed in chapter 1. First it defines the concepts of the proposed approach then the model design. Then the flow of the architecture approach is presented. Description of the evaluation method is also discussed in this chapter.

Chapter 5

5 Implementation

In this chapter proposed classification LSTM is implemented for game reviews. Systematic review has revealed that there are significant factors that can affect the model performance. First of all, dataset is very important in machine learning approaches. Because it has huge impact on the model performance. Dataset must be clean and managed according to the requirements. Therefore, analysis of dataset is done first to ensure it is applicable for training.

5.1. Results of Dataset Analysis

Before the implementation of LSTM model, we analyze the dataset to draw results and identify its shape. For this purpose, reviews are labelled into fourteen bugs categories which are proposed in j.Lewis taxonomy. [7]
The dataset contains 10,000 reviews of 15 games. These reviews are scrapped using the custom scrapper **SCRAPY.** Dataset also contain the information of review posted date, user name who posted the review and other metadata of the reviews. As shown in the blow table

Dataset	**No. of in-stances**	**No. of at-tributes**	**No. of target at-tributes**	**Missing values?**	**Pre-processing required?**
Ready or Not [46]	667	1	15	Yes	yes
Project Zombied [47]	667	1	15	Yes	Yes
Three Finger Battle Arena [48]	667	1	15	Yes	Yes
Baldur's Gate 3 [49]	667	1	15	Yes	Yes
Trip In Another World [50]	667	1	15	Yes	Yes
NEBULOUS: Fleet Command [51]	667	1	15	Yes	Yes
ERRANTE [52]	667	1	15	Yes	Yes
HITMAN 3 [53]	667	1	15	Yes	Yes
Grand Theft Auto V [54]	667	1	15	Yes	Yes

Tomb Raider [55]	667	1	15	Yes	Yes
Red Dead Redemption 2 [56]	667	1	15	Yes	Yes
Max Payne 3 [57]	667	1	15	Yes	Yes
Resident Evil 6 [58]	667	1	15	Yes	Yes
Just Cause 3 [59]	667	1	15	Yes	Yes
Star Wars: Battlefront 2 (Classic, 2005) [60]	667	1	15	Yes	Yes

Figure 11 Dataset Description

Total number of instances (reviews) in the whole dataset are fifteen thousand from 15 different games. Now, the number of dependent attribute in the dataset, used to perform analysis is one, the attribute of "text" (reviews). And the number of independent attributes are fifteen which are:

1. invalidPositionOverTime
2. implementationResponseIssue
3. invalidContextOverTime
4. interruptedEvent
5. invalidEventOccurraceOverTime
6. actionNotPossible
7. actionWhenNotAllowed
8. informationOutOfOrder
9. lackOfRequiredInformation
10. invalidInfoAccess
11. objectOutOfBoundForAnyState
12. objectOutOfBoundForSpecificState
13. artificialStupidity
14. invalidValueChange
15. invalidGraphicalRespresentation.

Furthermore, missing values are those reviews that are missed when crawled from the game steam engine, filled taking the most repeated value in the dataset. User reviews are plain text, words that use additional symbols or punctuation, no order of sentences, and are used by users to post comments about the app as shown in below figure 12.

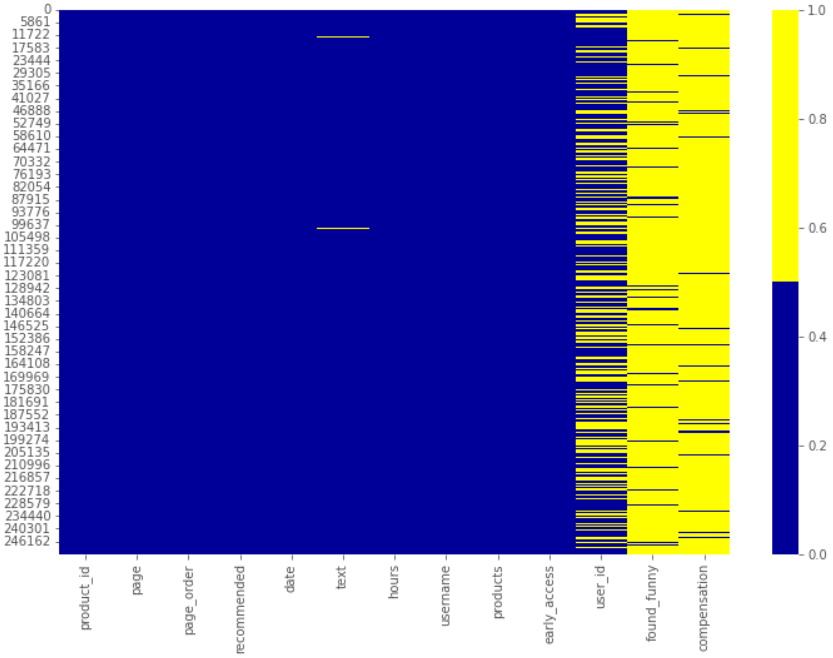


Figure 12 Missing Values Heat Map

The preparation of this raw text for analysis required pre-processing, including the removal of punctuation marks and symbols. By Pre-processing the dataset contain no special symbol no punctuation marks and no missing or null values. Figure 13 show the dataset after cleaning.

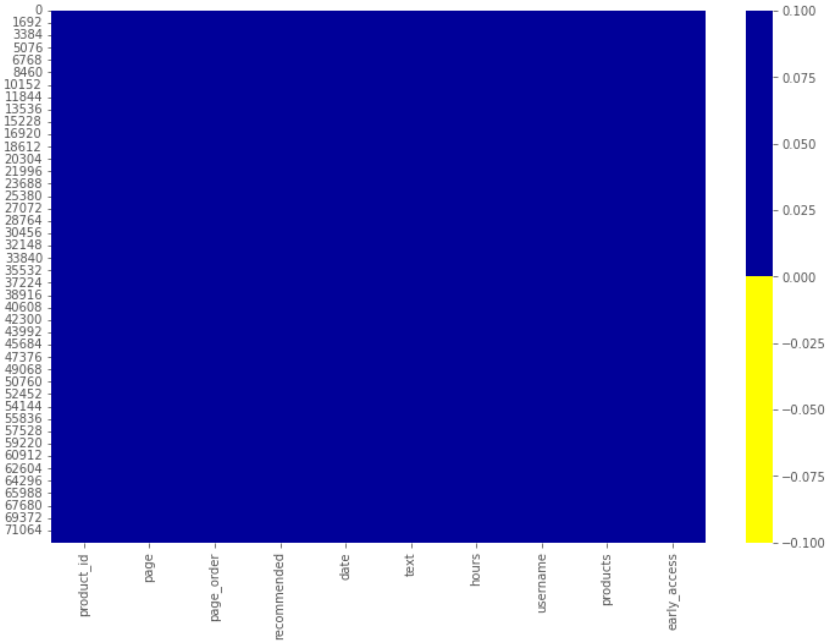


Figure 13 Dataset Heat map After Cleaning

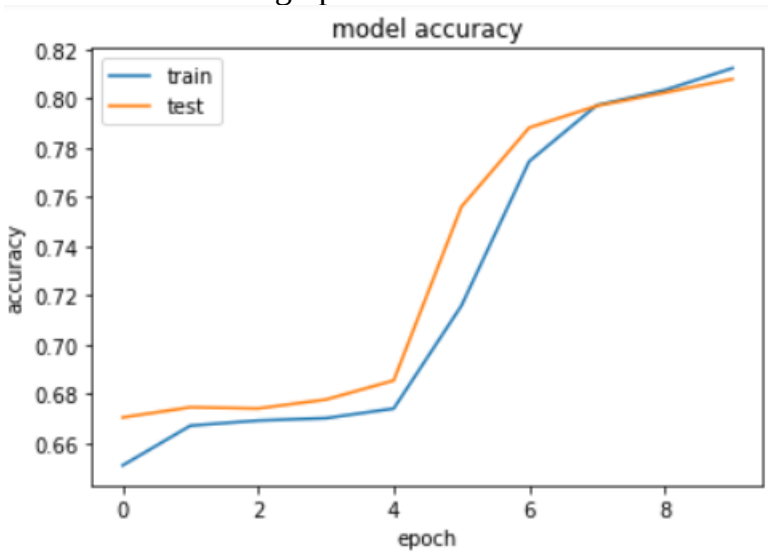
5.2. Results of LSTM Reviews Classifier

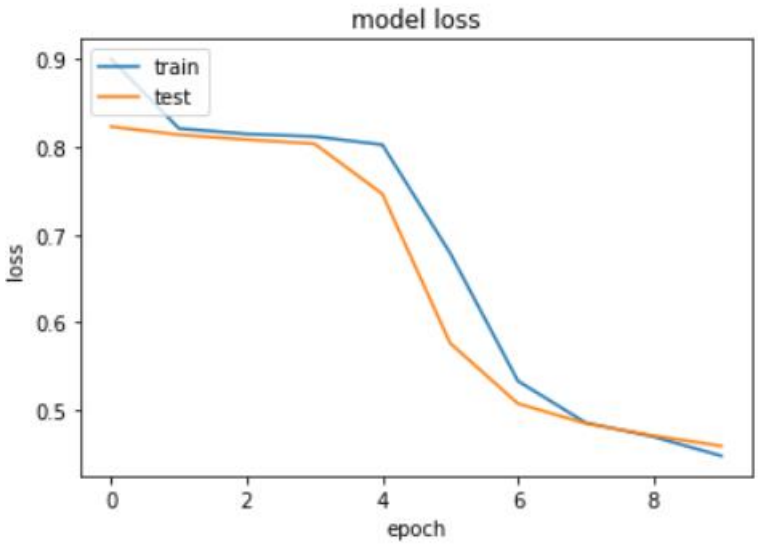
After dataset analysis, reviews text is cleaned as described above then Pre-Processing is applied using the best approaches that had applied before in literature. The dataset of user reviews is collected from fifteen games of different categories. Then ten thousand review are labeled manually by gaming experts.
To classify the reviews a deep learning LSTM model is designed which is trained over the ten thousand reviews dataset. This approach is defined in detail in chapter 4. Here we will discuss the results after implementing this model using python. Initially model is trained on 1500 reviews as described in pilot study in chapter 3. Later model is trained over 10,000 labelled reviews in which 8000 reviews are used for training and 2000 reviews are for validation. Detailed results of each epoch (hyper parameter that defines the number times that the learning algorithm will work through the entire training dataset) are shown in below table.

Epochs	Accuracy	Val-Accuracy
1	0.8234	0.8219
2	0.8237	0.8222
3	0.8344	0.8329
Average	0.8271	0.8256

Table 4 LSTM Model Accuracy on 10,000 reviews

For unseen data prediction accuracy was equal to one for the train model of LSTM. Model accuracy and loss is shown in below graphs.





5.3. Summary

This chapter implemented the proposed classification model for gaming reviews bugs classification name LSTM. At first analysis of dataset is done which show data representation before and after cleaning. Then the results of proposed classification model are presented along with the accuracy table. Evaluation metrics, accuracy is used and results are reported by averaging them over epochs using geometric mean average formula.

Chapter 6

6 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. But in order to meet the gamers expectations and increase the satisfaction of gamers it is much needed to detect multiple bugs from games. By their solution it can not only increase the quality of the games but also increase the number of users of the games.
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.

6.1. Research Contribution

This proposed multi-label classification model for gaming app reviews has developed to support both game developers and researchers. The gaming app bugs classification is a real world problem which gaming industry face on daily basis. Extraction of bugs from the reviews helps developers and gaming companies to make improvements in games. It helps them to get the information about the issues that gamers are facing while playing the games. User express more than one issue in a single review which require an automated model for multi-label classification of gaming bugs. This proposed research can be used for classification of gaming bugs from reviews.

6.2. 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.

Bibliography

[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] 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,” *2015 IEEE 31st Int. Conf. Softw. Maint. Evol. ICSME 2015 - Proc.*, no. 2, pp. 281–290, 2015, doi: 10.1109/ICSM.2015.7332474.

[22] R. Deocadez, R. Harrison, and D. Rodriguez, “Preliminary study on applying Semi-Supervised Learning to app store analysis,” *ACM Int. Conf. Proceeding Ser.*, vol. Part F1286, pp. 320–323, 2017, doi: 10.1145/3084226.3084285.

[23] Z. Kurtanovic and W. Maalej, “Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning,” *Proc. - 2017 IEEE 25th Int. Requir. Eng. Conf. RE 2017*, pp. 490–495, 2017, doi: 10.1109/RE.2017.82.

[24] H. X. Shi and X. J. Li, “A sentiment analysis model for hotel reviews based on supervised learning,” *Proc. - Int. Conf. Mach. Learn. Cybern.*, vol. 3, pp. 950–954, 2011, doi: 10.1109/ICMLC.2011.6016866.

[25] E. Oja, “Unsupervised learning in neural computation,” *Theor. Comput. Sci.*, vol. 287, no. 1, pp. 187–207, 2002, doi: 10.1016/S0304-3975(02)00160-3.

[26] M. Ahmad, S. Aftab, S. S. Muhammad, and S. Ahmad, “Machine Learning Techniques for Sentiment Analysis: A Review,” *Int. J. Multidiscip. Sci. Eng.*, vol. 8, no. 3, pp. 27–32, 2017, [Online]. Available: http://www.ijmse.org/Volume8/Issue3/paper5.pdf.

[27] L. M. Chiappe, “Thumbs up or thumbs down?: semantic orientation applied to unsupervised

classification of reviews. In ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pages 417–424, Morristown, NJ, USA, 2001. Ass," *Society*, vol. 12, no. 3, pp. 417–424, 2010, [Online]. Available: http://www.google.com.

[28] T. Zhang, M. Huang, and L. Zhao, "Learning structured representation for text classificationxvia reinforcement learning," *32nd AAAI Conf. Artif. Intell. AAAI 2018*, pp. 6053–6060, 2018.

[29] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requir. Eng.*, vol. 21, no. 3, pp. 311–331, 2016, doi: 10.1007/s00766-016-0251-9.

[30] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," *SN Comput. Sci.*, vol. 2, no. 3, 2021, doi: 10.1007/s42979-021-00592-x.

[31] C. N. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," *COLING 2014 - 25th Int. Conf. Comput. Linguist. Proc. COLING 2014 Tech. Pap.*, pp. 69–78, 2014.

[32] X. Zhang and Y. LeCun, "Text Understanding from Scratch," pp. 1–9, 2015, [Online]. Available: http://arxiv.org/abs/1502.01710.

[33] Y. Chen, "Convolutional Neural Network for Sentence Classification by," 2015.

[34] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," *NAACL HLT 2015 - 2015 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. Proc. Conf.*, pp. 103–112, 2015, doi: 10.3115/v1/n15-1011.

[35] A. Jacovi, O. S. Shalom, and Y. Goldberg, "Understanding Convolutional Neural Networks for Text Classification," *EMNLP 2018 - 2018 EMNLP Work. BlackboxNLP Anal. Interpret. Neural Networks NLP, Proc. 1st Work.*, pp. 56–65, 2018, doi: 10.18653/v1/w18-5408.

[36] B. Jang, M. Kim, G. Harerimana, S. U. Kang, and J. W. Kim, "Bi-LSTM model to increase accuracy in text classification: Combining word2vec CNN and attention mechanism," *Appl. Sci.*, vol. 10, no. 17, 2020, doi: 10.3390/app10175841.

[37] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep Learning Based Text Classification: A Comprehensive Review," vol. 1, no. 1, pp. 1–43, 2020, [Online]. Available: http://arxiv.org/abs/2004.03705.

[38] P. Liu, X. Qiu, and H. Xuanjing, "Recurrent neural network for text classification with multi-task learning," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2016-Janua, pp. 2873–2879, 2016.

[39] H. Qin and X. Sun, "Classifying bug reports into bugs and non-bugs using LSTM," *ACM Int. Conf. Proceeding Ser.*, pp. 16–19, 2018, doi: 10.1145/3275219.3275239.

[40] J. H. Wang, T. W. Liu, X. Luo, and L. Wang, "An LSTM approach to short text sentiment classification with word embeddings," *Proc. 30th Conf. Comput. Linguist. Speech Process. ROCLING 2018*, pp. 214–223, 2018.

[41] P. M. Vu, H. V Pham, T. T. Nguyen, and T. T. Nguyen, "Phrase-based extraction of user opinions in mobile app reviews," in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2016, pp. 726–731.

[42] M. Nayrolles and A. Hamou-Lhadj, “Towards a classification of bugs to facilitate software maintainability tasks,” *Proc. - Int. Conf. Softw. Eng.*, pp. 25–32, 2018, doi: 10.1145/3194095.3194101.

[43] W. Maalej and H. Nabil, “Bug report, feature request, or simply praise? On automatically classifying app reviews,” *2015 IEEE 23rd Int. Requir. Eng. Conf. RE 2015 - Proc.*, pp. 116–125, 2015, doi: 10.1109/RE.2015.7320414.

[44] A. Borrelli, V. Nardone, G. A. Di Lucca, G. Canfora, and M. Di Penta, “Detecting Video Game-Specific Bad Smells in Unity Projects,” *Proc. - 2020 IEEE/ACM 17th Int. Conf. Min. Softw. Repos. MSR 2020*, pp. 198–208, 2020, doi: 10.1145/3379597.3387454.

[45] S. M. Liu and J. H. Chen, “A multi-label classification based approach for sentiment classification,” *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1083–1093, 2015, doi: 10.1016/j.eswa.2014.08.036.

[46] “No Title,” [Online]. Available: http://steamcommunity.com/app/1144200/reviews/?browsefilter=mostrecent&p=1.

[47] “Project Zombied.” http://steamcommunity.com/app/108600/reviews/?browsefilter=mostrecent&p=1.

[48] “Three Finger Battle Arena,” [Online]. Available: http://steamcommunity.com/app/108600/reviews/?browsefilter=mostrecent&p=1.

[49] “Baldur’s Gate 3.” http://steamcommunity.com/app/1893860/reviews/?browsefilter=mostrecent&p=1.

[50] “Trip In Another World.” http://steamcommunity.com/app/1826960/reviews/?browsefilter=mostrecent&p=1.

[51] “NEBULOUS: Fleet Command.” http://steamcommunity.com/app/887570/reviews/?browsefilter=mostrecent&p=1.

[52] “ERRANTE,” [Online]. Available: http://steamcommunity.com/app/1871830/reviews/?browsefilter=mostrecent&p=1.

[53] “HITMAN 3,” [Online]. Available: http://steamcommunity.com/app/1659040/reviews/?browsefilter=mostrecent&p=1.

[54] “Grand Theft Auto V.” http://steamcommunity.com/app/271590/reviews/?browsefilter=mostrecent&p=1%0A.

[55] “Tomb Raider.” http://steamcommunity.com/app/203160/reviews/?browsefilter=mostrecent&p=1.

[56] “Red Dead Redemption 2.” http://steamcommunity.com/app/1174180/reviews/?browsefilter=mostrecent&p=1.

[57] “Max Payne 3.” http://steamcommunity.com/app/204100/reviews/?browsefilter=mostrecent&p=1.

[58] “Resident Evil 6.” http://steamcommunity.com/app/221040/reviews/?browsefilter=mostrecent&p=1.

[59] “Just Cause 3.” http://steamcommunity.com/app/225540/reviews/?browsefilter=mostrecent&p=1.

[60] “Star Wars: Battlefront 2 (Classic, 2005),” [Online]. Available: http://steamcommunity.com/app/6060/reviews/?browsefilter=mostrecent&p=1.