Empirical Software Engineering (2019) 24:4006–4033
https://doi.org/10.1007/s10664-019-09733-6

Identifying gameplay videos that exhibit bugs in computer games

Dayi Lin[1] · **Cor-Paul Bezemer**[2] · **Ahmed E. Hassan**[1]

Published online: 25 June 2019

Abstract

With the rapid growing market and competition in the gaming industry, it is challenging to develop a successful game, making the quality of games very important. To improve the quality of games, developers commonly use gamer-submitted bug reports to locate bugs in games. Recently, gameplay videos have become popular in the gaming community. A few of these videos showcase a bug, offering developers a new opportunity to collect context-rich bug information. In this paper, we investigate whether videos that showcase a bug can automatically be identified from the metadata of gameplay videos that are readily available online. Such bug videos could then be used as a supplemental source of bug information for game developers. We studied the number of gameplay videos on the Steam platform, one of the most popular digital game distribution platforms, and the difficulty of identifying bug videos from these gameplay videos. We show that naïve approaches such as using keywords to search for bug videos are time-consuming and imprecise. We propose an approach which uses a random forest classifier to rank gameplay videos based on their likelihood of being a bug video. Our proposed approach achieves a precision that is 43% higher than that of the naïve keyword searching approach on a manually labelled dataset of 96 videos. In addition, by evaluating 1,400 videos that are identified by our approach as bug videos, we calculated that our approach has both a mean average precision at 10 and a mean average precision at 100 of 0.91. Our study demonstrates that it is feasible to automatically identify gameplay videos that showcase a bug.

Keywords Gameplay videos · Bug report · Computer games · Steam

Communicated by: Emerson Murphy-Hill

✉ Dayi Lin
dayi.lin@cs.queensu.ca

Cor-Paul Bezemer
bezemer@ualberta.ca

Ahmed E. Hassan
ahmed@cs.queensu.ca

[1] Software Analysis and Intelligence Lab (SAIL), Queen's University, Kingston, ON, Canada

[2] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada

1 Introduction

The gaming industry is a rapidly growing industry with a predicted market of over 90 billion U.S. dollars by 2020 (PwC 2016). In this competitive market, creating a successful game is an expensive and daunting task. It is necessary to invest a tremendous amount of effort to deliver high quality games, resulting in a *crunch culture* which pressures game developers into working long hours (Schreier 2018; Phillips 2018).

Despite the large efforts that are spent on the development process of a game, it is not possible to discover or fix all bugs in a game before releasing it to the market. Our prior work shows that 80% of the Steam games release urgent updates to fix issues such as feature malfunctions or game crashes (Lin et al. 2017). A common practice to ensure the user-perceived quality of a game is to allow gamers to submit bug reports that describe any encountered bugs during gameplay. Many studios employ discussion forums or specific features in their games for gamers to report bugs (e.g., Blizzard (Sixen 2010), EVE Online (Habakuk 2017), League of Legends (Afic 2015)). The discussion forums and in-game bug reporting features usually require gamers to provide a title for the bug, a detailed description of the bug, and the necessary steps to reproduce the bug. Unfortunately, to learn about bugs from bug reports, game developers are dependent on gamers to submit such reports.

One possibility to learn more about the problems that gamers encounter is by studying gameplay videos (Lewis et al. 2010). In recent years, gameplay videos have become popular in the gaming community. Two of the top five YouTube channels with the highest number of worldwide subscribers are related to gaming (Ekaterina and Gross 2017). The Steam platform, one of the most popular online distribution platforms for games (Lin et al. 2019b), also allows gamers to share a link to and discuss their YouTube gameplay videos on the Steam Community website (Valve 2018a). Gamers share gameplay videos that cover several topics including game tutorials, game play-throughs (i.e., a video of playing a game from start to finish), and game bugs. These bug videos open up a new opportunity for developers to collect information about bugs. Several studios have realized the value of bug videos, and encourage gamers to submit relevant screenshots or videos along with their bug report (Sixen 2010; Afic 2015; Habakuk 2017), especially for bugs in the game's graphics ("should probably always have a screenshot" (Sixen 2010)) and general functionality bugs (which "may only be visible in movies" (Sixen 2010)).

In this paper, we conduct an in-depth study of gameplay videos that are posted by gamers. In addition, we study how bug videos can be identified automatically. We first conducted a preliminary study of 1,785,366 gameplay videos on the Steam platform, to understand the number of gameplay videos and the difficulty of distinguishing bug videos from other gameplay videos. We found that 21.4% of the games on the Steam platform receive more than 50 game videos, and up to a median of 13 hours of video runtime per day, and that a naïve keyword searching approach for identifying videos that showcase a game bug only achieves a precision of 56.25%. To help developers better leverage such bug videos, this paper proposes an approach that uses a random forest classifier to further rank the videos with their likelihood of showcasing a game bug, using features that are extracted from the metadata of videos (such as their length). Our evaluation shows that our proposed approach achieves a precision of 0.80, which is 43% higher than that of the naïve keyword searching approach, on a manually labelled dataset of 96 videos. In addition, by evaluating 1,400 videos that are identified by our approach as bug videos, we calculated that our proposed

approach has both a mean average precision at 10 and a mean average precision at 100 of 0.91. Our study makes the following contributions:

- The first approach that showcases how to automatically identify bug videos with high precision.
- An analysis of the characteristics of videos that showcase a game bug.
- A labelled dataset and the metadata of 1,496 bug videos (Lin et al. 2019a) to enable further studies in this important research direction.

Paper Organization The rest of this paper is organized as follows. Section 2 provides a brief description of the Steam platform and YouTube Gaming channels. Section 3 presents our methodology. Section 4 presents the results of our preliminary study. Section 5 describes our proposed approach to determine the likelihood of a gameplay video showcasing a game bug. Section 6 discusses our results and gives directions for future work. Section 7 discusses related work. Section 8 discusses the threats to validity of our study. Finally, Section 9 concludes the paper.

2 Background

In this section, we briefly describe YouTube and YouTube Gaming, and the Steam platform.

2.1 YouTube and YouTube Gaming

YouTube is one of Google's subsidiaries, and serves as the largest video-sharing website and the second-most popular website in the world (Alexa 2018). YouTube allows users to view, upload, comment, rate, and share videos on various topics, such as video blogging, music videos, educational videos, and gameplay videos. When uploading a video to YouTube, the user will provide the title, description, keywords, and category of the video. YouTube provides a pre-defined set of categories for users to choose from, such as "Sports", "Education", and "Gaming". If a user selects "Gaming" as the category of the video, YouTube would ask the user to enter the title of the game that is associated with the video.

Videos on YouTube are aggregated into channels. Each user has their own YouTube channel, which contains the videos that they uploaded. In addition, YouTube automatically aggregates videos into system-generated channels, such as YouTube Gaming channels (YouTube 2018). YouTube Gaming[1] is a site under YouTube that is dedicated to gaming videos and channels. When a user uploads a video under the "Gaming" category, and provides the title of the associated game, YouTube automatically adds the video into the YouTube-generated gaming channel for that game, and labels the viewing page of the video with the title of the game, with a link to the YouTube Gaming channel of the game. However, unlike user-owned channels, the system-generated channels for specific games do not provide an exhaustive list of all the videos under the channels.

[1] https://gaming.youtube.com/

2.2 The Steam Platform

Steam is a PC game distribution platform that is developed by Valve Corporation, and is considered as one of the largest digital distribution platforms for PC games (Lin et al. 2019b), with more than 23,000 games available (Valve 2018b). The Steam platform has two major components: the Steam Store (Valve 2018b), and the Steam Community (Valve 2018a), which provide digital rights management (DRM) and social networking services respectively.

The Steam Store allows a gamer to purchase games or activate games that are purchased through third-party vendors. Such games are stored in the gamer's Steam library, and can be launched through the Steam client.

In addition, the Steam Community provides gamers with social network-like features, including friends lists, game reviews, discussion forums, and markets for in-game items. The Steam Community also allows gamers to share audiovisual content, such as screenshots and videos, under each game's community page. It is worth noting that the Steam platform does not provide a video storage service. To post a video for a game on the Steam Community, a gamer needs to upload the video to YouTube first, then authorize the Steam platform to access the gamer's YouTube account for the list of potential videos for posting. The gamer can then select videos that they wish to share, and associate the videos with games. The videos are then posted on the gamer's profile page on Steam. Gamers are allowed to edit the title and the description of a video post, which are filled by default with the video's YouTube title and description.

3 Methodology

In this section, we describe our methodology. First, we describe how we extracted and processed our data. In general, we collected metadata of videos for games on the Steam platform (i.e., Steam games) and for games that are not on the Steam platform (i.e., non-Steam games). Table 1 presents the description of our collected dataset. Figure 1 gives an overview of our data collection process. Second, we describe the statistical and classification techniques that we use throughout our study.

3.1 Collecting Videos for Steam Games

To collect videos for Steam games, we first collected a list of all Steam games from the Steam Store. We took a snapshot of all the 16,252 games that were available in the Steam Store on August 1st, 2017, using a customized crawler (i.e., a tool that automatically collects the required data).

We then developed a customized crawler that collects video posts for each Steam game in the Steam Community. For each Steam game in the collected game list, the crawler visited

Table 1 Gameplay video dataset description

# of studied Steam games	16,252
# of unique video posts in the Steam Community	1,785,366
# of Steam videos that are available on YouTube	1,636,689
# of non-Steam games for verification	4
# of YouTube videos for non-Steam games	20,754

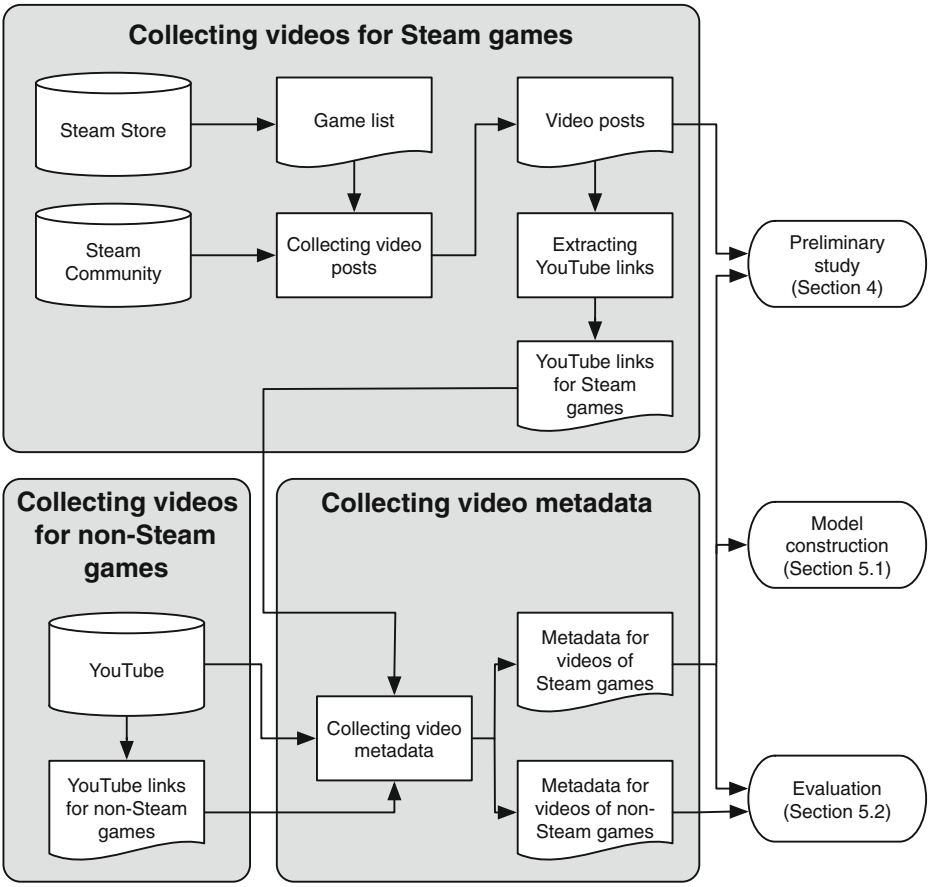

Fig. 1 Overview of the data collection process for our study of gameplay videos

its Steam Community page, and collected all the video posts on the Steam Community for that game. Specifically, the crawler collected the following information for each video post: title, description, and the YouTube link of the video. The crawler also recorded the specific Steam game to which a video post belongs. Table 2 gives an example of a video post.

In total, we collected 1,989,140 video posts on the Steam Community for all the games. We noticed that there were some video posts with the same YouTube links (i.e., different video posts for the same video). We observed that the video posts with duplicated YouTube links are posted by the same Steam accounts under the same game, potentially by mistake. We removed such duplicate video posts. We collected a total of 1,785,366 unique YouTube links for videos of Steam games.

Table 2 An example of a video post

Title	Fallout New Vegas Hardest punch
Description	I hit him so hard he broke
YouTube link	https://youtu.be/-TiTf5G4wpg
Associated Steam game	Fallout: New Vegas[1]

[1] https://store.steampowered.com/app/22380/

3.2 Collecting Videos for Non-steam Games

To evaluate the generalizability of our results, we used a customized crawler to collect the YouTube videos for games that were not released on the Steam platform. This crawler is available in the supplementary material of our paper (Lin et al. 2019a). We focused on games that were published by Electronic Arts (EA), as many recent AAA games[2] from EA were exclusively released on EA's own online distribution platform for PC games, the Origin platform (Electronic Arts Inc 2018) (and hence not on Steam). We selected 4 EA Origin exclusive games that have a large number of gameplay videos on YouTube, i.e., FIFA 16, FIFA 17, NHL 16, and NHL 17. As the YouTube Gaming channels do not provide a comprehensive list of all videos under a game's channel, the crawler first used the search function provided by YouTube to search for videos with keywords being the title of the game, then it visited the page of each video in the search result to confirm whether the video is associated with the target game. In total, we collected 20,754 videos for the 4 studied games that are not on the Steam platform.

3.3 Collecting Video Metadata

For each YouTube video collected in Sections 3.1 and 3.2, we used a customized crawler to collect the video's metadata on YouTube. In particular, we collected the title, category, description, tags, and length (in seconds) for each video.

We noticed that 148,677 YouTube links from the Steam Community were not accessible on YouTube at the time of crawling, due to the following reasons:

- The uploader deleted the video
- The video violated copyright from a third party company
- The video is being processed (e.g., transcoded)
- The video is private
- The video violated YouTube's Terms of Service
- The video is not available in the country from which our crawler ran (i.e., Canada)
- The YouTube account that is associated with the video has been terminated

In total, we collected the metadata for 1,657,443 YouTube videos (1,636,689 for Steam games and 20,754 for non-Steam games), which were uploaded by 323,325 Steam accounts. For the videos of the Steam games, we linked each video's metadata from YouTube to the specific video post on the Steam Community.

To the best of our knowledge, there is no explicit rule in Steam's Terms of Service and Copyright that forbids the crawling of publicly available pages on the Steam Store/Steam Community. We also did not crawl any sub-urls that are explicitly disallowed by Steam's robots.txt. All information from YouTube was collected through their API, following their regulations about data collection.

3.4 Used Classification Techniques

In our study, we use a classifier to determine the likelihood that a game video showcases a bug in a game. In Section 4, we use a keyword search "classifier" as a naïve approach for identifying bug videos. In Section 5, we compare the performance of the logistic regression,

[2]Games with the highest budgets for development and promotion.

neural network and random forest classifiers for identifying bug videos. We explain the concept of the classifiers that we used below.

- **Keyword search:** A naïve classifier that searches for the occurrence of certain keywords in the video metadata.
- **Logistic regression:** A logistic regression classifier captures the relation between one or more independent variables (i.e., the metadata) and one dependent variable (i.e., bug video or non-bug video) using a polynomial expression and a sigmoid function.
- **Neural network:** A neural network classifier mimics the brain at a small scale. A neural network consists of an input and output layer with multiple hidden layers in between. The hidden layers contain neurons, which are connected to each other through weighted connections. During the supervised training of the network, the weights on the connections are optimized for the classification task for which the network is trained. A neural network decides on the value of the dependent variable by following the connections between the input and output layers based on the input data and the connection weights.
- **Random forest** (Ho 1995)**:** A random forest consists of multiple decision trees that are based on randomly-selected subsets of the training data and features. A decision tree traverses a tree structure from the root to a leaf to make a decision based on the input data and the conditions in the tree. Each leaf node in a decision tree corresponds to a value of the dependent variable. A random forest decides on the value of the dependent variable by taking a majority vote across the decision trees in the forest.

3.5 Used Statistical Techniques

In our study we use the Wilcoxon rank sum test to compare distributions of values. The Wilcoxon rank sum test is an unpaired, non-parametric statistical test of which the null hypothesis is that two input distributions are identical. We use a non-parametric test as it does not require the input distributions to be normally distributed. We use a threshold of 0.05 for the Wilcoxon rank sum test's p-value to decide whether the null hypothesis is rejected. If the p-value of the test is smaller than 0.05, the null hypothesis is rejected and we conclude that the distributions are significantly different.

While the Wilcoxon rank sum test indicates whether two distributions are significantly different, it does not quantify the magnitude of the difference. To further quantify the magnitude of the difference of the two distributions, we computed Cliff's delta d (Long et al. 2003) effect size. We used the following threshold for interpreting d, as proposed by Romano et al. (2006):

$$\text{Effect size} = \begin{cases} negligible(N), & \text{if } |d| \leq 0.147. \\ small(S), & \text{if } 0.147 < |d| \leq 0.33. \\ medium(M), & \text{if } 0.33 < |d| \leq 0.474. \\ large(L), & \text{if } 0.474 < |d| \leq 1. \end{cases}$$

For example, a Cliff's delta of 0.1 indicates that while the difference between two distributions may be significant, it has a negligible magnitude (and may not be that relevant in practice). In contrast, a Cliff's delta of 0.5 indicates that the magnitude of the difference is large.

4 Preliminary Study of Gameplay Videos on the Steam Platform

As there exists no prior research on mining gameplay videos from the Steam platform, in this preliminary study, we aim at providing an overview of such videos. In addition, as

shown in prior work (Lewis et al. 2010), gameplay videos contain valuable information for game developers, such as game bugs. In our preliminary study, we examined the effort that is needed for developers to identify gameplay videos that showcase a game bug (i.e., *bug videos*) on the Steam platform.

Approach: We examined the effort that is required from developers to identify bug videos using two naïve approaches:

1. **Watch all videos of a game.** To identify bug videos from the pool of all posted videos of a game on the Steam platform, the most accurate way is to manually watch all the videos of a game. As the bug may show up at the end of the video, it is necessary to watch the whole video to locate the bug. Hence, we calculated the number of videos and the accumulated video length for each game, to understand the effort that would be needed by developers to watch all the videos of their games.
2. **Search for keywords.** Another naïve approach to identify bug videos is to search for the occurrence of certain keywords in the video metadata (e.g., in the title, description, and tags). Hence, we applied a naïve keyword searching approach to game videos to evaluate its performance, and to understand if such an approach would be effective for developers.

We consider the problem of identifying bug videos as a binary classification problem, i.e., we classify all the videos of a game into two classes: bug videos (True), or not bug videos (False). Hence, we have the confusion matrix as shown in Table 3.

The precision of an approach to identify bug videos can be calculated as $\frac{TP}{TP+FP}$. We use precision instead of recall to evaluate the performance of an approach because we consider bug videos as a supplemental source of bug reports (in addition to textual issue reports). Therefore, we prefer missing actual bug videos over wrongly identifying videos as bug videos, so that developers would not waste time watching non-bug videos.

There are five text fields in the metadata we collected for videos that are used for the keyword search: the Steam video post title and description, and the YouTube title, description, and tags. Note that the Steam and YouTube title and description are not always the same.

To find suitable keywords to search for bug videos, we used the following process:

1. **Removing stop words.** Stop words (e.g., "the") have a high occurrence in the studied text and are irrelevant to the study. We removed the stop words from the text fields in the metadata using the stop word list provided by the Natural Language Toolkit (NLTK) (NLTK Project 2017). The NLTK provides a collection of tools that can be used to preprocess natural language data.
2. **Stemming.** To avoid different forms of the same word (e.g., "argue", "argued", "argues", "arguing") being recognized as different words, we stemmed (e.g., "argu") all the words in the text fields in the metadata using NLTK. As the stop word list from NLTK is not stemmed, we apply stemming after removing the stop words.

Table 3 Confusion matrix for identifying bug videos

		Predicted	
		Bug videos	Not bug videos
Actual	Bug videos	True Positive (TP)	False Negative (FN)
	Not bug videos	False Positive (FP)	True Negative (TN)

Table 4 Selected keywords for matching

Groups	Keywords (# of videos)	Description
Bug	bug (32,166), glitch (28,025)	Directly related to bugs
Hack	hack (28,459), hacker (8,405), cheat (21,639), cheater (6,336)	Hackers / cheaters in a game make use of game bugs

3. **Removing uncommon words.** Words occurring in a very small number of videos may be game-specific and hence not helpful for identifying bug videos across all games. For each remaining word, we counted the number of videos that contain the word in their metadata. Then we kept the words that occur in more than 1,636 videos (0.1% of the number of collected videos that are available on YouTube).
4. **Selecting bug-related keywords.** We manually checked each of the words and selected a list of keywords that are related to bugs in games. Table 4 shows the keywords that we selected (lower case, singular form).

When selecting the keywords, we noticed an interesting keyword "wtf", which occurs in 16,649 videos' metadata. "Wtf" is a unique label for gameplay videos that record surprising moments in the gameplay, either an unexpected event or an impressive operation. We did not include "wtf" in our keyword list, as we observed that it is very noisy for identifying game bug videos. However, it may be interesting for developers to include this keyword, if their resources for watching these videos permit to do so. The "hack" and "cheat" keywords are often used to indicate a loophole in the game. As we show in our prior work (Lin et al. 2017), loopholes are one of the main reasons for releasing an urgent update (to fix the loophole) and hence are considered bugs by game developers.

Findings: **21.4% of the studied games have more than 50 videos on the Steam platform.** Figure 2 shows the distribution of the number of videos for each game. Games on the Steam platform have a median of 9 videos. 3,483 (21.4%) games have each received more than 50 game videos. The game with the most game videos on the Steam platform is the *Counter-Strike: Global Offensive* game, for which 352,443 game videos were posted.

26.2% of the games on the Steam platform received an accumulated length of more than 10 hours of game video, with the longest accumulated length for a particular game reaching over 3 years. Figure 3 shows the distribution of the accumulated length of game videos for each game. Games on the Steam platform received a median accumulated length of 136 minutes (2.3 hours) of game videos. We calculated that 3,026 (26.2%) games receive an accumulated length of more than 10 hours, and 1,926 (16.7%) games receive an accumulated length of more than 24 hours (1 day) of game videos. The longest accumulated

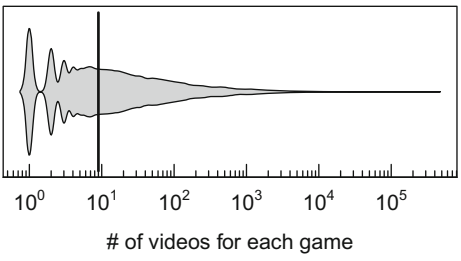

Fig. 2 The distribution of the number of videos of each game. The vertical line shows the median value of the distribution

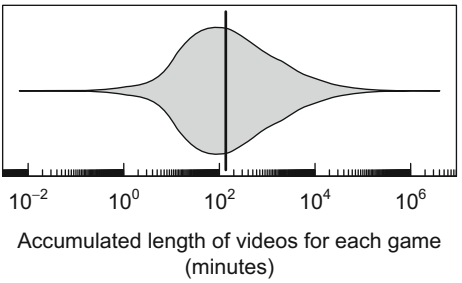

Fig. 3 The distribution of the accumulated length of videos of each game. The vertical line shows the median value of the distribution

length of game videos that was received by a single game is 1,593,005 minutes (3.03 years – for the *Counter-Strike: Global Offensive* game).

To further demonstrate the workload for developers to watch gameplay videos everyday, we calculated the accumulated length of videos for each game per day. Figure 4 shows the distribution of the median accumulated length of videos per day. The median value of the distribution is 16.38 minutes, with a maximum of 13.27 hours, suggesting that in general developers of a game will receive around 16 minutes of videos in a day; and in extreme cases, developers can receive up to 13 hours of gameplay videos on median in a day.

Using keywords to identify bug videos has a precision of 56.25%. We applied the keyword searching approach to all 1,636,689 gameplay videos for Steam games that are available on YouTube. In total, the keywords matched 83,321 videos. We randomly selected a statistically representative sample of 96 videos from the videos that matched the keywords (with 95% confidence level and 10% confidence interval), and manually verified whether they are bug videos, by watching the full video. The total runtime of the 96 videos in the sample is 14.97 hours, which underlines the value of automatically identifying bug videos. In total, 54 out of these 96 videos are bug videos. Hence, the precision of a naïve keyword searching approach for identifying bug videos is 56.25%. We manually checked the false positives to understand the reasons behind the low precision of the keyword searching approach, and we extracted the following cases for the false positives:

- Advertising other bug videos under the same channel in the description
- Stuffing irrelevant keywords (Google 2018) in the description that include our keywords
- Videos about games that themselves involve the keywords (e.g. *Age of Barbarian*: a "hack and slash" game;[3] *Bug Butcher*: a game about catching bugs[4])

The majority of videos of most games are posted by a few Steam accounts, while each account only posts videos for a few games. For each game, we sum up the number of video posts that are posted by the top 3 Steam accounts with the highest number of video posts, and calculated the ratio of the sum to the total number of video posts the game received. Figure 5 shows the distribution of the ratio.

As shown in Fig. 5, games receive a median of 75% of videos from their top 3 Steam accounts. We further calculated the number of games for which each account posted videos. We found that 81% of the Steam accounts only posts videos of one game, and 96% of the accounts posts videos of no more than 5 games.

We compared the ratio between the games with at least 100 videos (which is 14% of the studied games), and the games with less than 100 videos. Figure 6 shows that the top

[3] https://www.youtube.com/watch?v=W9in3AEYuJk
[4] https://www.youtube.com/watch?v=pnFuzq8Ii8o

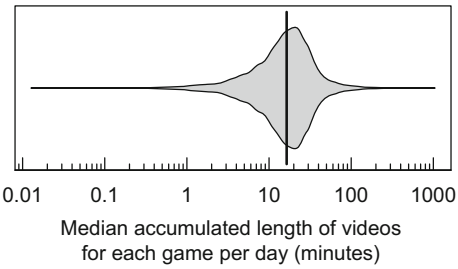

Fig. 4 The distribution of the accumulated length of videos of each game per day. The vertical line shows the median value of the distribution

3 Steam accounts contributed a median of 83% of the videos of a game with less than 100 videos, and a median of 29% of the videos of a game with at least 100 videos. In addition, we applied the Wilcoxon rank sum test (Wilcoxon 1945) on the two distributions. As the p-value of the test is smaller than 0.05, we concluded that the ratio of the games with at least 100 videos and the games with less than 100 videos are significantly different. We calculated that the difference of these two distributions has a large Cliff's Delta effect size.

The *GameGlitches* account contributes the most videos to the Steam Community. Figure 7 shows the empirical cumulative distribution function of the number of videos contributed by each account. 62% of the Steam accounts posted only one video to the Steam Community. We calculated that 92% of the Steam accounts posted at most 10 videos. The account with the highest number of videos is *GameGlitches*,[5] with 4,413 videos. The majority of the videos under the *GameGlitches* account focuses on the glitches of games.

> ***Summary:*** *Game videos may contain important information for game developers (e.g., showcases of a bug). However, 21.4% of the games on the Steam platform receive more than 50 game videos in total, and up to a median of 13 hours of video runtime per day. Hence, manually watching through gameplay videos daily requires a considerable amount of resources. In addition, using a naïve keyword searching approach to identify bug videos only has a precision of 56.25%. Hence, a more advanced approach is needed to identify bug videos. In addition, it may be worthwhile for game developers to monitor the videos that are posted by certain Steam accounts.*

5 Determining the Likelihood that a Gameplay Video Showcases a Game Bug

In our preliminary study, we showed that naïve approaches do not perform well in identifying bug videos. To help game developers leverage the valuable information in gameplay videos such as showcases of bugs in the games, in this section, we study whether more advanced classifiers can improve the performance of the naïve keyword searching approach.

We train logistic regression, neural network and random forest classifiers to determine the likelihood that a game video showcases a bug of a game, based on the results of the keyword searching approach. We then evaluate the performance of these three classifiers and select the best-performing one (the random forest classifier) for a deeper investigation.

[5] http://steamcommunity.com/id/GamesGlitches/

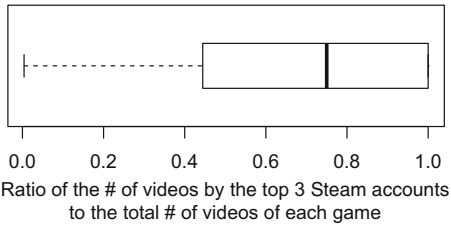

Fig. 5 The distribution of the ratio of the number of videos by the top 3 Steam accounts to the total number of videos of each game

In particular, we evaluate our random forest classifier on a larger data set and we use the classifier to understand which factors are correlated with a higher likelihood of a game video showcasing a game bug. By automatically determining the likelihood that a game video showcases a game bug, developers can watch game videos that have a high likelihood and avoid wasting time on videos that do not showcase a game bug. Figure 8 shows an overview of our approach.

5.1 Selecting the Best-Performing Classifier for Identifying Bug Videos

5.1.1 Approach

To train the classifiers, we used factors that measure the contents of video posts and the metadata of the YouTube video. Table 5 defines each factor and the rationale behind our choice of that factor. We used the identified keywords in our preliminary study (i.e., "bug", "glitch", "hack", "hacker", "cheat", "cheater") for features that concern keywords in this section. We did not include context factors (e.g., the game that the video is about) to ensure that our approach can be adapted by newly-released games. In addition, we did not include factors that are related to the account that uploads the video (e.g., the number of videos that were uploaded) to ensure that our approach works for videos from both newly-created and old accounts.

We built logistic regression, neural network and random forest classifiers to determine the likelihood that a game video showcases a game bug. These three classifiers represent three families of classifiers (respectively generalized linear models, neural nets and decision trees). We conducted two steps (C1, C2) to construct the classifiers, and one step (A1) to analyze the constructed classifiers. We conducted two additional steps (A2 and A3) to further analyze the best-performing classifier (i.e., the random forest).

Step C1 - Removing Correlated and Redundant Factors Correlated factors in a model can lead to the misinterpretation of factor importance (Tantithamthavorn and Hassan 2018). To

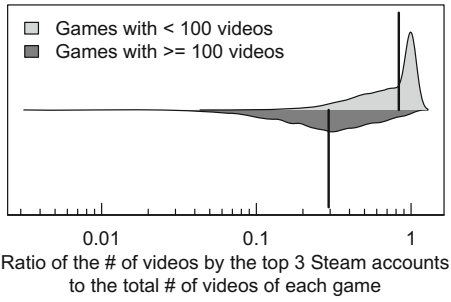

Fig. 6 The distribution of the ratio of the number of videos by the top 3 Steam accounts to the total number of videos of games with less than 100 videos and at least 100 videos

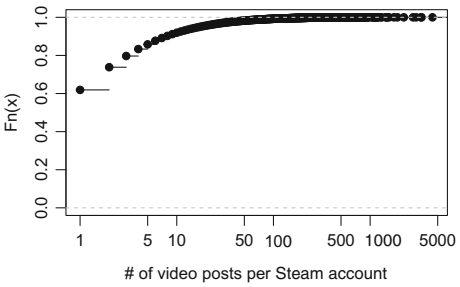

Fig. 7 The empirical cumulative distribution function (Fn(x)) of the number of video posts per Steam account

mitigate correlated factors, we used the Spearman rank correlation to calculate how strongly two factors are correlated. If two factors have a Spearman $|\rho| > 0.7$, we remove one of them from our model. We used Spearman rank correlation because we observed our data to be skewed, and Spearman rank correlation does not assume a normal distribution of the data. Figure 9 shows the hierarchically clustered Spearman $|\rho|$ values.

We found that the corresponding factors across YouTube and Steam (e.g., video post title length and YouTube title length) are all highly correlated. As stated in Section 2, although the video posts on the Steam platform are hosted on YouTube, Steam allows gamers to customize the title and description of the video posts. We found that most of the video post titles are the same as their YouTube titles. In fact, only 167,486 (1.0%) video posts have different titles compared to their YouTube titles. We manually looked into the videos with different titles for the video post and on YouTube, and identified the following common reasons:

- Steam filtered inappropriate words, while YouTube did not
- Gamers did not enter a title for the Steam video post
- Gamers added a prefix to their YouTube titles (e.g. their YouTube channel name)

In addition, only 192,998 (1.2%) videos have different descriptions on Steam and on YouTube. The most common differences are:

- Copyright disclaimers for the background music or other part of the videos in the YouTube description
- Stuffed keywords at the end of the YouTube description
- Descriptions in different languages

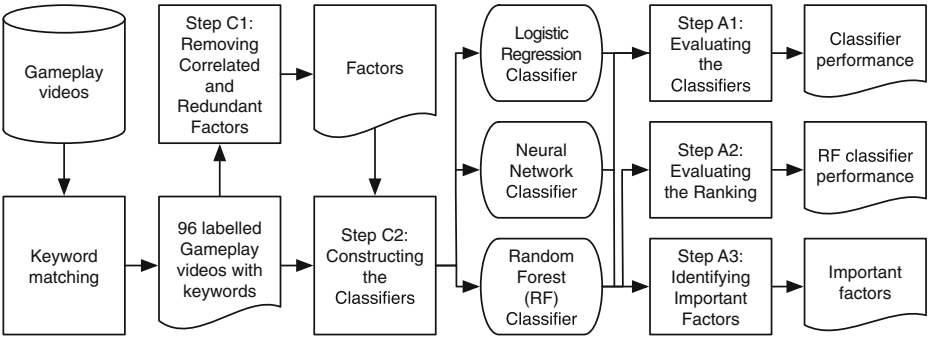

Fig. 8 Overview of the approach to determine the likelihood that a gameplay video showcases a game bug

Table 5 Factors used in the built classifiers

Factor	Description (d) - Rationale (r)
Video length	d: The length of the video in seconds. r: Bug videos could be shorter, as they may only show the bug instead of a full gameplay session.
Video post title length	d: The number of characters in the title of the video post. r: A longer title of the video post could contain a description of a game bug.
Video post description length	d: The number of characters in the description of the video post. r: A long description could contain stuffed keywords (Google 2018), which are usually irrelevant to the content of the video and hence decrease the likelihood of the video showcasing a bug.
YouTube title length	d: The number of characters in the YouTube title. r: A longer YouTube title could contain a description of a game bug.
YouTube description length	d: The number of characters in the YouTube description. r: A long YouTube description could contain stuffed keywords (Google 2018), which are usually irrelevant to the content of the video and hence decrease the likelihood of the video showcasing a bug.
# of YouTube tags	d: The number of YouTube tags. r: A video with a large number of YouTube tags could contain stuffed keywords (Google 2018), which are usually irrelevant to the content of the video and hence decrease the likelihood of the video showcasing a bug.
# of keyword matches in video post title	d: The number of occurrences of the keywords in the title of the video post. r: A video post with more keyword matches in the title could have a higher likelihood of being a bug video.
# of keyword matches in video post description	d: The number of occurrences of the keywords in the description of the video post. r: A video post with more keyword matches in the description could have a higher likelihood of being a bug video.
# of keyword matches in YouTube title	d: The number of occurrences of the keywords in the YouTube title. r: A video with more keyword matches in the YouTube title could have a higher likelihood of being a bug video.
# of keyword matches in YouTube description	d: The number of occurrences of the keywords in the YouTube description. r: A video with more keyword matches in the YouTube description could have a higher likelihood of being a bug video.
# of keyword matches in YouTube tags	d: The number of occurrences of the keywords in the YouTube tags. r: A video with more keyword matches in YouTube tags could have a higher likelihood of being a bug video.

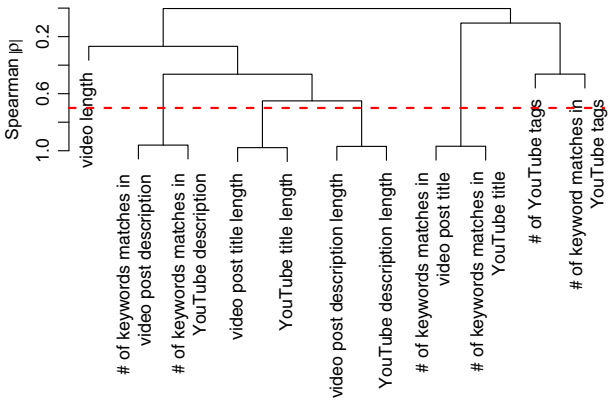

Fig. 9 The hierarchically clustered Spearman $|\rho|$ values of factors. The dotted red line represents $|\rho| = 0.7$, our threshold for deciding whether two factors are strongly correlated

To ensure that our approach can be applied to game videos that are not on the Steam platform, we kept the factors from the YouTube metadata, and removed the video post title length, # of keywords matches in the video post title, video post description length and the # of keywords matches in the video post description from our classifier.

To further remove redundant factors (i.e., factors that can be explained by a combination of other factors), we performed a redundancy analysis using the `redun` function in the `Hmisc` package in R. We found that after removing the correlated factors, there are no redundant factors.

Step C2 - Constructing the Classifiers We used the manually labelled data set of 96 videos from Section 4 to train the classifiers. We used the following parameters for the classifiers:

- **Logistic regression**: We used the default parameters for the logistic regression classifier.
- **Neural network**: We employed a grid search to find the following optimal parameters for our neural network classifier: a single hidden layer with 6 nodes and a decay of 0.1.
- **Random forest**: We used a default value of 3 for the `mtry` parameter (the number of variables that are randomly sampled as candidates at each split).

Step A1 - Evaluating the Classifiers To compare the performance of the classifiers to the naïve keyword search approach and each other, we used the out-of-sample bootstrap validation technique (Tantithamthavorn et al. 2017). The out-of-sample bootstrap validation technique consists of two steps:

1. For the given original dataset of size N, we randomly draw a bootstrap sample of size N with replacement.
2. We train classifiers using the bootstrap sample, and test the classifiers using the rows that are not drawn in the first step. As the bootstrap sample is drawn with replacement, on average 36.8% of the rows will not be drawn in the first step (Efron 1983).

In our study, N is 96, since we are using the manually labelled 96 videos to train our classifier. The out-of-sample bootstrap process was repeated 1,000 times, and the average out-of-sample precision and AUC were calculated.

Table 6 The performance of the studied classifiers on the labelled data set of 96 videos

Classifier	AUC	Precision
Keyword search	-*	0.56
Logistic regression	0.79	0.79
Neural network	0.70	0.70
Random forest	0.84	0.80

*As the keyword search does not return probabilities its AUC cannot be calculated

5.1.2 Result

Our random forest classifier achieves the best performance. Table 6 shows the performance of the classifiers. The out-of-sample bootstrap validation shows that on average, the Random Forest classifier has an AUC of 0.84, and a precision of 0.80 on the manually labelled data, which is higher than the precision of the naïve keyword approach (0.56, see Section 4) and the other classifiers. The performance of the logistic regression classifier comes closest to that of the random forest. The performance of the neural network classifier is considerably lower. We expect that the used data set is too small for the neural network to perform well. However, it is important to keep in mind that manually labelling videos is extremely time-consuming. Therefore, an important requirement of a classifier for identifying bug videos is that it performs well after being trained with a small training data set. Hence, in the rest of the paper, we focus on the random forest classifier.

5.2 A deeper Investigation of the Random Forest Classifier

5.2.1 Step A2 - Evaluating the Provided Ranking by the Random Forest Classifier

As we aim at using the random forest classifier to provide a list of videos ranked by their likelihood of being a bug video, only using precision and AUC to evaluate the performance of the approach will ignore the order in which the bug videos are presented. Hence, in addition to the out-of-sample bootstrap validation, we further quantify the performance of our approach using the following set of metrics:

Precision at n ($P@n$) (Larson 2010) is commonly used to evaluate the performance of an information retrieval system. For our approach, we have $P@n = \frac{r}{n}$, where r is the number of actual bug videos at rank n.

Average Precision at n ($AP@n$) (Larson 2010) is a variant of the well-known average precision measure. While precision at n ignores the positions of the result, average precision at n takes into account both the number of bug videos in the top n and the positions of those bug videos. The average precision at n is defined as:

$$AP@n = \frac{\sum_{i=1}^{n} rel(i) \times P@i}{NF}$$

Where $rel(i) = 1$ if the i^{th} video in the ranked list is a bug video, and $rel(i) = 0$ otherwise, and NF is the normalization factor. Although it is common to normalize by the total number of bug videos, here we use the number of correctly identified bug videos as the normalization factor, as proposed by Baeza-Yates et al. (1999). This normalization is called "Average Precision at Seen Relevant Documents", and it avoids the problem in precision-oriented evaluation, where one may not have judged enough documents to know the recall. In the remainder of the paper, we use $AP@n$ to refer to Average Precision at Seen Relevant Documents.

Table 7 Games that are used for the evaluation of our approach

Title	# of videos	# of videos identified by keyword matching	# of videos identified by random forest classifier
Steam games			
Counter-Strike: Global Offensive	352,443	12,495	9,376
Dota 2	65,023	841	650
Garry's Mod	37,245	945	617
Arma 3	32,049	825	507
Grand Theft Auto V	31,127	2,310	1,386
Counter-Strike	26,377	728	537
DayZ	20,234	1,590	968
Rocket League	19,759	264	166
Counter-Strike: Source	17,691	1,046	574
ARK: Survival Evolved	17,080	639	198
Non-Steam games			
FIFA 16	5,701	396	161
FIFA 17	6,008	464	200
NHL 16	4,624	400	226
NHL 17	4,421	382	205

Mean Average Precision at n ($MAP@n$) (Larson 2010) is the arithmetic mean of all average precision at n.

We selected 10 games from the Steam platform with the highest number of videos, and calculated the metrics for each of the games. We also selected 4 games that were not released on the Steam platform, and collected their game videos on the YouTube platform, to evaluate the generalizability of our approach. Table 7 shows the games we used for our evaluation.

For each of the games, we calculated the $P@n$ and $AP@n$ for $n = 100$. Hence, we manually verified whether 1,400 game videos showcase a bug. In addition, to simulate the process of developers using our proposed approach as they go through pages of results, with each page displaying 10 videos, we calculated the $AP@10$ of each page of results for each game. Lastly, we calculated the $MAP@n$ for $n = 100$, for the Steam games and non-Steam games respectively.

5.2.2 Step A3 - Identifying the Important Factors

To understand the importance of each factor in a random forest classifier, we calculated the mean decrease accuracy (Louppe et al. 2013) for each factor. The mean decrease accuracy is commonly used in prior studies for measuring factor importance of random forest classifiers (Kabinna et al. 2018) and is calculated as follows: for each tree in the random forest, the error rate on the out-of-bag portion of the data is recorded, and compared with the error rate after permuting each factor. The bigger the difference is, the more important that factor is deemed. We used the `importance` function in the `RandomForest` package in `R` to calculate the mean decrease accuracy.

Table 8 Result of the evaluation

Title	P@100	AP@100	AP@10									
			1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80	81-90	91-100
Steam games												
Counter-Strike: Global Offensive	0.83	0.84	0.84	0.81	0.94	0.91	1.00	0.58	1.00	0.77	0.79	0.89
Dota 2	0.93	0.96	1.00	1.00	0.88	1.00	1.00	0.98	0.88	1.00	0.73	0.96
Garry's Mod	0.74	0.81	0.98	0.75	0.96	0.56	0.96	0.99	0.73	0.73	0.46	0.74
Arma 3	0.93	0.97	1.00	1.00	1.00	1.00	0.92	0.91	0.99	1.00	0.98	0.84
Grand Theft Auto V	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Counter-Strike	0.89	0.85	0.54	1.00	0.83	1.00	1.00	0.99	1.00	0.91	1.00	0.93
DayZ	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.96	1.00	1.00
Rocket League	0.65	0.90	1.00	1.00	1.00	0.95	0.91	0.29	0.53	0.34	0.33	0.91
Counter-Strike: Source	0.72	0.83	0.75	0.96	0.98	0.96	0.82	1.00	0.81	0.92	0.50	0.11
ARK: Survival Evolved	0.92	0.97	1.00	1.00	1.00	0.88	1.00	0.75	1.00	1.00	0.57	0.93
MAP@100 for Steam games		0.91										
Non-Steam games												
FIFA 16	0.91	0.88	0.70	1.00	1.00	1.00	1.00	0.93	1.00	0.75	1.00	0.97
FIFA 17	0.96	0.96	0.99	1.00	1.00	0.88	1.00	1.00	1.00	1.00	0.96	0.96
NHL 16	0.90	0.99	1.00	1.00	1.00	1.00	0.95	1.00	1.00	1.00	0.55	0.46
NHL 17	0.89	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.90	0.76	0.44
MAP@100 for non-Steam games		0.91										

5.2.3 Results

Our random forest classifier achieves a mean average precision at 100 of 0.91. We manually checked the 100 gameplay videos with the highest likelihood of showcasing a game bug for the 14 games for evaluation. In total, we manually checked 1,400 gameplay videos. Table 8 shows the performance metrics for each of the evaluated games.

Table 8 shows that for both the 10 Steam games and the 4 non-Steam games that are used for evaluation, the classifier obtained a mean average precision at 100 of 0.91. For the *Garry's Mod* game, the *Rocket League* game and the *Counter-Strike: Source* game, we observed a relatively low $P@100$ (below 0.8). However, the $AP@100$ for these three games are all higher than their $P@100$ (all above 0.8, up to 0.25 higher than their $P@100$), suggesting that our random forest classifier is able to assign a higher likelihood of showcasing a bug to bug videos than to non-bug videos.

To further understand the false positive cases, we manually checked the 168 videos that were wrongly identified by our classifier as bug videos. The false positive cases were mainly introduced by the ambiguity of keywords, specifically, the keyword "hack". We noticed that in the *Counter-Strike* community (i.e., the *Counter-Strike* game, the *Counter-Strike: Source* game, and the *Counter-Strike: Global Offensive* game), based on context, the term "hack" can be used to describe gameplay that is so skilled that it looks like hacking (e.g., http://youtu.be/RG78hWtY5Mo). Another common reason was that gamers' usernames contain "hack" while the videos are not about hacking. In addition, we also noticed cases in which the videos were about hacked gamer accounts. We calculated that by excluding the keyword "hack" and "cheat", 23% of the false positive cases would not be labelled as bug videos. Hence, developers can choose to exclude the keyword "hack" to further increase the precision of our approach, if desired. Other reasons for the false positive cases include videos about how to patch a bug, gamers using the word "buggy" to refer to motor vehicles, gamers' usernames containing keywords, and keyword stuffing.

The average precision at 10 of each page of results remains high, even after already having seen 10 pages of results. Figure 10 shows the relationship between $AP@10$ and the page of results, with each page containing 10 videos. Even for page 10, the classifier obtains an average $AP@10$ of 0.80, and a $MAP@10$ of 0.92. Although more study may be

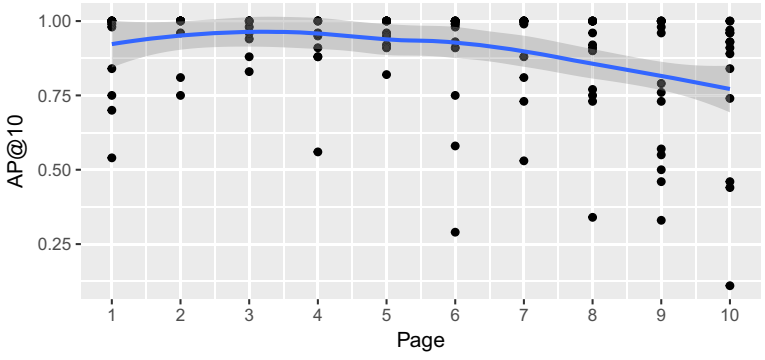

Fig. 10 $AP@10$ against the page of results (with 10 videos per page). The blue lines are fitted using Local Polynomial Regression (Cleveland and Devlin 1988). The grey areas show the approximate 95% confidence bands

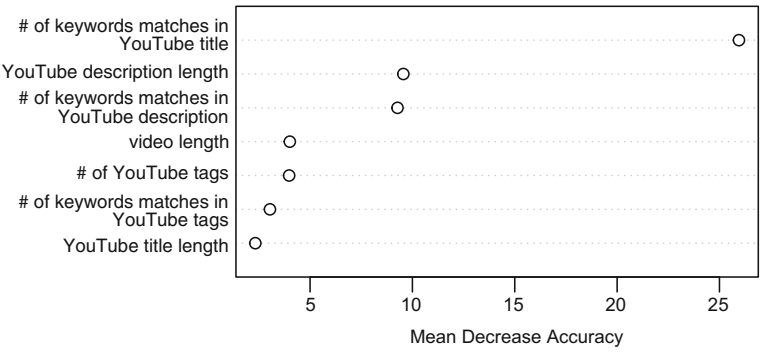

Fig. 11 The mean decrease accuracy for each factor in the classifier

needed to evaluate how well the classifier performs after 10 pages, the classifier still has a high performance even after having seen 10 pages of results already.

The most important factor for identifying bug videos is having a keyword in the YouTube title. Figure 11 shows the mean decrease accuracy of each factor that is used in our random forest classifier.

As Fig. 11 shows, the number of keyword matches in YouTube title has the highest mean decrease accuracy, with the length of the YouTube description, and the number of keyword matches in the YouTube description being the second and the third respectively. Hence, the number of keyword matches in the YouTube title is the most important factor for identifying bug videos.

To understand how changes in the three most important factors' values affect the likelihood of a game video being a bug video, we plotted the predicted likelihood against the three aforementioned factors. Figure 12 shows the impact of the three most important factors on the likelihood of a game video being a bug video.

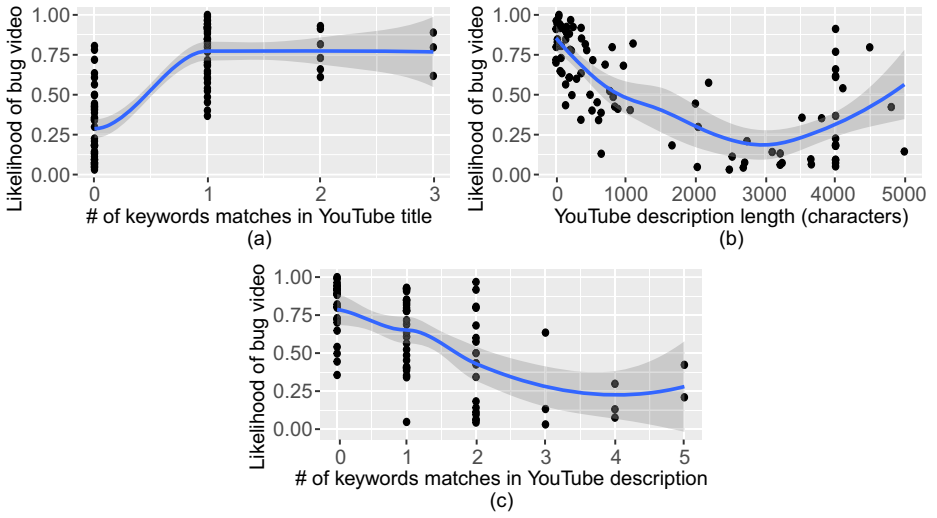

Fig. 12 The predicted likelihood of bug videos against the three most important factors for deciding the likelihood of a video being a bug video. The blue lines are fitted using Local Polynomial Regression (Cleveland and Devlin 1988). The grey areas show the approximate 95% confidence bands

For the number of keyword matches in the YouTube title, Fig. 12a shows that the likelihood of the video showcasing a bug is higher when there is at least one occurrence of a keyword, but the likelihood does not go higher as the occurrence increases. For the number of keywords matches in the YouTube description, Fig. 12c shows that the likelihood decreases as the occurrence of keywords in YouTube description increases. A possible explanation is that, as stated in Section 4, people may stuff irrelevant keywords in the videos' YouTube descriptions as an attempt of search engine optimization. In addition, Fig. 12b shows that the likelihood of a video being a bug video is higher when the length is either very short or very long.

Summary: *The proposed random forest classifier achieves a mean average precision at 100 of 0.91, and a precision (0.80) that is 43% higher than the naïve keyword searching approach (0.56). Videos with at least one occurrence of a keyword (i.e., "bug", "glitch", "hack", "hacker", "cheat", "cheater") in their YouTube title, a shorter YouTube description, and lower occurrence of keywords in their YouTube description, are more likely to be bug videos.*

6 Discussion

Nowadays, gameplay videos are extremely popular, as demonstrated by the popularity of the Twitch[6] service for broadcasting gameplay sessions. Prior work showcases the usefulness of gameplay videos for game developers. For example, Lewis et al. (2010) showed that there exist many videos that exhibit a bug in a game. However, to identify these bug videos, developers would have to watch through all gameplay videos of their games. In this paper we showed that it is possible to automatically identify bug videos, solely based on their metadata. Our random forest approach can be trained with a small data set of labelled videos, achieving a mean average precision of 0.91 at 100. Moreover, our approach is not game-specific and as a result, it can be applied to new games, or games with relatively few gameplay videos as well.

An interesting question is how useful these bug videos are to game developers. One issue is that bug videos usually do not explicitly describe the steps to reproduce a bug, which is essential for developers who wish to fix that bug (Zimmermann et al. 2010). It is important to realize that while these videos may not lead directly to bug fixes, it does allow developers to learn more about the occurrence of bugs. In some cases these bugs may be completely unknown, and in these cases the developer could reach out to the poster of the video to acquire more details about the bug (or even, to ask the video poster to submit an actual bug report to help fix the bug). In other cases, developers may already know about the bugs but chose not to fix them. In these cases, bug videos could help to see more instantiations of the same bug, making it easier to reproduce or fix hard-to-fix bugs.

The research presented in this paper is the first important step towards automatically extracting useful information for game developers from gameplay videos. In the remainder of this section, we discuss several directions for future work.

[6] https://www.twitch.tv/

6.1 Future Work Direction 1: Leveraging Gameplay Video Contents to Identify Useful Videos for Game Developers

The presented approach in this paper is solely based on the metadata of gameplay videos. However, in many cases this metadata is either not available or not meaningful. For example, a play-through video, which is much longer than most of the bug videos studied in this paper, could exhibit one or more bugs in a game. However, it is less likely that these bugs are identified explicitly in the video's metadata. In such cases, it would be helpful to identify the exact bug video segments by analyzing the contents of a gameplay video. We envision several research challenges in this direction:

- **Automatically distinguishing buggy behaviour from regular game behaviour in a gameplay video.** Future work could leverage existing work on extracting game logs automatically from gameplay videos (Jacob et al. 2014; Luo et al. 2018) by mining such logs for anomalies which could represent bugs.
- **Precisely pinpointing the start and end times of the bug video segment.** Simply knowing that a video exhibits a bug is not helpful if the video is long, as it would still require the game developer to watch the full video. Instead, future work should investigate how to precisely pinpoint the start and end times of a bug video segment in a longer video (such as a Twitch playing session), possibly through the analysis of the spoken words in the video.
- **Identifying other types of useful videos for game developers.** We focused on bug videos, but there exist many other types of gameplay videos that are useful for game developers. For example, game developers could leverage gameplay videos to identify performance issues or confusing storylines of the game.

6.2 Future Work Direction 2: Automatically Appending Textual Bug Reports with Bug Videos

Although most bug videos do not explicitly describe the steps to reproduce a bug, a bug video can clarify the way in which a bug manifests itself. Therefore, bug videos can clarify existing textual bug reports. Future work should investigate how bug videos can be linked automatically with existing textual bug reports. For example, future work could combine prior work in the game understanding field (e.g., by Luo et al. 2018) to identify the game objects in a bug video and map these objects to bug reports.

7 Related Work

In this section, we discuss prior work that is related to our study. Rather than providing an exhaustive overview of all studies that leverage gameplay videos, which is outside the scope of our paper, we discuss the most important studies. The contribution of our study in comparison to prior work is that to the best of our knowledge, we are the first to propose an approach for identifying game videos that showcase a game bug.

7.1 Studies on Gameplay Videos

Lewis et al. (2010) explored the software engineering aspect of gameplay videos. Lewis et al. summarized a taxonomy of video game bugs by observing patterns from bug videos on

YouTube. They identified YouTube's bug videos as *"a rich resource ... provide a startling amount of coverage; far more than any single research group could ever hope to expose personally"*.

Another usage of gameplay videos that is relevant to software engineers is to automatically extract game logs. These game logs can be used by game designers to study the field usage of a game or the field behaviour of its players. For example, Jacob et al. (2014) used image processing techniques and predefined image patterns to extract game logs from gameplay videos of the *Super Mario Bros.* game. Luo et al. (2018) used convolutional neural networks and transfer learning to extract game logs from gameplay videos of the *Super Mario Bros.*, *Megaman* and *Skyrim* games.

Several studies investigated why people watch gameplay videos. For example, Sjöblom and Hamari (2017) and Hilvert-Bruce et al. (2018) found that Twitch users watch Twitch videos for different types of gratification, such as social integrative reasons and tension release.

Several studies use gameplay videos in combination with machine learning to learn how to play a game. For example, Mnih et al. (2013) used gameplay videos as the input to a convolutional neural network to learn how to play Atari games. Mnih et al.'s study laid the foundation for the automated gameplay of more advanced games such as the *Doom* game (Kempka et al. 2016). Guzdial et al. (2017) proposed a search-based approach to learn the game engine of a game from gameplay videos. In another study, Guzdial et al. (2018) propose a framework for automatically creating commentaries for gameplay videos. Fulda et al. (2018) automatically detect interaction modes in gameplay videos, i.e., situations that require a certain interaction. Their approach recommends whether the identified objects in a screenshot are a threat or require exploration, bartering or solving a puzzle (such as picking a lock).

Finally, several studies focus on using gameplay videos to automatically generate game levels. Guzdial and Riedl (2016) designed an unsupervised machine learning process that can automatically generate full video game levels from gameplay videos, using K-Means and probabilistic graphical models. Summerville et al. (2016) proposed a machine learning technique that uses Long Short-Term Memory Recurrent Neural Networks (LSTM RNNs) to generate levels based on latent play styles learnt from the Super Mario Bros gameplay videos.

7.2 Studies on Improving Bug Report Quality

Several studies have focused on the quality of bug reports. Zimmermann et al. (2010) conducted a survey among the community of Apache, Eclipse and Mozilla to understand the characteristics of high quality bug reports, and revealed the mismatch of information between what developers need and what bug reporters provide. The study highlights that for developers, the most needed information is the steps to reproduce, which is considered by users as difficult to provide. In addition, the most severe problem is not wrong information, but absent information. The study also highlights the potential value of video bug reports, using the "Best of Bugzilla" bug report, Eclipse bug 113206 as an example. This bug report includes a video to demonstrate the complicated steps to reproduce the bug. Hooimeijer and Weimer (2007) analyzed over 27,000 bug reports for Mozilla Firefox to model the bug report quality. The analysis showed that attachment count and comment count have the highest impact on the quality of a bug report. Linstead and Baldi (2009) proposed a new measure of bug report quality called latent topic coherence, by modelling Gnome bug reports with Latent Dirichlet Allocation (LDA). Zimmermann et al. (2009) proposed four broad directions for enhancing the current bug tracking systems to collect bug reports with higher quality, one of which suggests the bug reporting tool to integrate capture/replay functionality or a screenshot.

7.3 Mining Online Gaming Distribution Platforms

Mining data from online gaming distribution platforms (such as the Steam platform) is an area that has been actively researched recently. Chambers et al. (2005) collected various game-related data from both servers of games and the Steam platform. The study demonstrated that it is difficult to provide enough resources at launch time of a game. In addition, the study also claimed that gamers are extremely difficult to satisfy. Blackburn et al. (2014) analyzed more than 12 million player profiles on the Steam Community, with 700,000 profiles being flagged as cheaters. They showed that whether a player has cheating friends plays an important role in whether a player becomes a cheater.

In our prior work, we analyzed the Steam platform to study multiple aspects of Software Engineering practice in game development (Lin et al. 2017, 2018, 2019b). We studied urgent updates of popular games on the Steam platform to help developers avoid unnecessary stress during game development, and produce games with higher user-perceived quality (Lin et al. 2017). One of our major findings is that games using frequent update strategy tend to require more urgent updates. In addition, we analyzed the early access games on the Steam platform (Lin et al. 2018), and suggested game developers to use the early access model as a way to build up positive reputations, and avoid relying on the sales during early access stage for game development. Recently, we conducted a large-scale empirical study of the game reviews on the Steam platform (Lin et al. 2019b). We concluded that game reviews are different from mobile app reviews along several aspects, and that prior studies on mobile app reviews may need to be revisited.

8 Threats to Validity

This section presents the threats to the validity of our findings.

8.1 Internal Validity

A threat to the validity of our findings is that we only collected videos that are explicitly linked to specific games, on both the Steam platform, and YouTube. This constraint ensures that the number of videos of a game is an accurate lower bound. However, as it is not mandatory for gamers to link videos to games in either the Steam platform or on YouTube, there may exist more videos related to studied games that were not collected.

In both Sections 4 (96 videos) and 5 (1,400 videos), we manually verified whether a gameplay video is a bug video, for a total of 1,496 gameplay videos. While this process was manual, it was straightforward (though time-consuming). We would like to emphasize that our model is necessary to reduce the required time to identify bug videos, not to reduce the difficulty of the identification process. Because of the low degree of difficulty of the identification, we did not cross-validate the identification process.

In this paper, we built a universal model to predict the likelihood that a gameplay video showcases a game bug across all games. However, the gamer community of different games may have different patterns when posting bug videos. In this case, game-specific models may outperform a universal model. In practice, newly-released games may not have enough historical data to train a game-specific model. Nonetheless, our approach can be adopted for building game-specific models as well.

It is possible that in some rare instances a person has multiple Steam accounts. Hence, the reported numbers in our findings are a high estimate of the actual number of gamers for which those findings hold.

8.2 External Validity

In our preliminary study (Section 4), we focused on the gameplay videos on the Steam platform. The findings of our preliminary study may not be generalizable to other games on different distribution platforms. However, as stated in Section 2, Steam is the largest online distribution platform for PC games. Hence, the games on the Steam platform are representative for a large number of games. Future studies are necessary to investigate how our results apply to games from other platforms, such as the Xbox.

In Section 5, we evaluated the generalizability of our approach using four games that are not on the Steam platform. Future studies are needed to examine the generalizability of the approach on a larger number of games.

9 Conclusion

The user-perceived quality is crucial to the success of a game. Hence, it is necessary to efficiently and effectively deal with bugs in a game.

In recent years gameplay videos have become popular in the gaming community to showcase the problems of a game to other gamers. However, there have been no studies of how developers can leverage such gameplay videos as an additional source for learning about bugs in their games.

In this paper, we explored the practicality of using gameplay videos that are available online as a supplementary source of learning about bugs in games. We first conducted a preliminary study on gameplay videos on the Steam platform, and found that naïve approaches to identify bug videos, such as keyword searching, are inefficient and imprecise. In particular, the naïve keyword searching approach only has a precision of 56.25%. We then proposed an approach that uses the metadata of gameplay videos to train logistic regression, neural network and random forest classifiers, to determine the likelihood that a gameplay video showcases a game bug. We further evaluated the best-performing classifier, the random forest classifier, on 10 Steam games and 4 non-Steam games. Overall, the approach achieves both a mean average precision at 10 and a mean average precision at 100 of 0.91, and shows a good generalizability. We also identified the impact of different factors on the likelihood of a video related to bugs. Our study makes the following contributions:

- The first to showcase that it is practical to automatically identify bug videos with high precision.
- An analysis of the characteristics of bug videos.
- A labelled dataset and the metadata of 1,496 bug videos (Lin et al. 2019a) to enable further studies in this important research direction.

Our paper takes the first important step to automatically identify gameplay videos that contain useful information for game developers. In future work, researchers should extend our approach to identify useful information in longer gameplay videos. For example, a future approach could pinpoint the exact locations of useful information in long gameplay

videos on the Twitch service, a service for broadcasting gameplay sessions. The challenge of identifying useful information in Twitch gameplay sessions is that (1) they are often several hours long and (2) they contain less metadata than the videos that we studied in this paper. Another future direction is to automatically append existing textual bug reports with video snippets in which the particular bug is exhibited.

References

Afic R (2015) How to report a game bug. https://boards.na.leagueoflegends.com/en/c/bug-report/3mQGBEjA-how-to-report-a-game-bug, (last visited: Mar 25 2019)

Alexa (2018) Youtube.com traffic, demographics and competitors - alexa. https://www.alexa.com/siteinfo/youtube.com, (last visited: Mar 25 2019)

Baeza-Yates R, Ribeiro-Neto B et al (1999) Modern information retrieval, vol 463. ACM Press, New York

Blackburn J, Kourtellis N, Skvoretz J, Ripeanu M, Iamnitchi A (2014) Cheating in online games: a social network perspective. ACM Trans Internet Technol (TOIT) 13(3):9

Habakuk CCP (2017) Bug reporting - eve community. https://community.eveonline.com/support/test-servers/bug-reporting/, (last visited: Mar 25 2019)

Chambers C, Feng Wc, Sahu S, Saha D (2005) Measurement-based characterization of a collection of online games. In: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement, USENIX Association, pp 1–14

Cleveland WS, Devlin SJ (1988) Locally weighted regression: an approach to regression analy sis by local fitting. J Amer Stat Assoc 83(403):596–610

Efron B (1983) Estimating the error rate of a prediction rule: improvement on cross-validation. J Amer Stat Assoc 78(382):316–331

Ekaterina P, Gross N (2017) 4 reasons people watch gaming content on Youtube. https://www.thinkwithgoogle.com/consumer-insights/statistics-youtube-gaming-content/ (last visited: Mar 25 2019)

Electronic Arts Inc (2018) Origin. https://www.origin.com, (last visited: Mar 25 2019)

Fulda N, Ricks D, Murdoch B, Wingate D (2018) Threat, explore, barter, puzzle: A semantically-informed algorithm for extracting interaction modes. In: The Workshops of the 32nd AAAI Conference on Artificial Intelligence, pp 1–5

Google (2018) Irrelevant keywords - search console help. https://support.google.com/webmasters/answer/66358, (last visited: Mar 25 2019)

Guzdial M, Riedl M (2016) Game level generation from gameplay videos. In: Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference

Guzdial M, Li B, Riedl MO (2017) Game engine learning from video. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI), pp 3707–3713

Guzdial M, Shah S, Riedl M (2018) Towards automated let's play commentary. CoRR arXiv:1809.09424

Hilvert-Bruce Z, Neill JT, Sjöblom M, Hamari J (2018) Social motivations of live-streaming viewer engagement on Twitch. Comput Hum Behav 84:58–67

Ho TK (1995) Random decision forests. In: Proceedings of the 3rd International Conference on Document Analysis and Recognition. IEEE, vol 1, pp 278–282

Hooimeijer P, Weimer W (2007) Modeling bug report quality. In: Proceedings of the 22nd IEEE/ACM international conference on Automated software engineering, ACM, pp 34–43

Jacob LB, Kohwalter TC, Machado AFV, Clua EWG, d Oliveira D (2014) A non-intrusive approach for 2d platform game design analysis based on provenance data extracted from game streaming. In: Brazilian Symposium on Computer Games and Digital Entertainment, pp 41–50

Kabinna S, Bezemer CP, Shang W, Syer MD, Hassan AE (2018) Examining the stability of logging statements. Empir Softw Eng 23(1):290–333

Kempka M, Wydmuch M, Runc G, Toczek J, Jaśkowski W (2016) ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In: IEEE Conference on Computational Intelligence and Games (CIG), pp 1–8

Larson RR (2010) Introduction to information retrieval. J Amer Soc Inf Sci Technol 61(4):852–853

Lewis C, Whitehead J, Wardrip-Fruin N (2010) What went wrong: a taxonomy of video game bugs. In: Proceedings of the 5th International Conference on the Foundations of Digital Games, ACM, pp 108–115
Lin D, Bezemer CP, Hassan AE (2017) Studying the urgent updates of popular games on the Steam platform. Empir Softw Eng 22(4):2095–2126
Lin D, Bezemer CP, Hassan AE (2018) An empirical study of early access games on the Steam platform. Empir Softw Eng 23(2):771–799
Lin D, Bezemer CP, Hassan AE (2019a) Supplementary material for our paper. https://github.com/SAILResearch/suppmaterial-19-dayi-game_video, (last visited: Mar 25 2019)
Lin D, Bezemer CP, Zou Y, Hassan AE (2019b) An empirical study of game reviews on the Steam platform. Empir Softw Eng 24(1):170–207
Linstead E, Baldi P (2009) Mining the coherence of GNOME bug reports with statistical topic models. In: 6th IEEE International Working Conference on Mining Software Repositories. IEEE, pp 99–102
Long JD, Feng D, Cliff N (2003) Ordinal Analysis of Behavioral Data. Wiley, New York
Louppe G, Wehenkel L, Sutera A, Geurts P (2013) Understanding variable importances in forests of randomized trees. In: Advances in Neural Information Processing Systems, pp 431–439
Luo Z, Guzdial M, Liao N, Riedl M (2018) Player experience extraction from gameplay video. In: AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, pp 1–7
Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller MA (2013) Playing Atari with deep reinforcement learning. CoRR arXiv:1312.5602
NLTK Project (2017) Natural Language Toolkit. https://www.nltk.org/, (last visited: Mar 25, 2019)
Phillips T (2018) The human cost of Red Dead Redemption, 2. https://www.eurogamer.net/articles/2018-10-25-the-human-cost-of-red-dead-redemption-2, (last visited: Mar 25 2019)
PwC (2016) Value of the global video games market from 2011 to 2020 (in billion u.s. dollars). https://www.statista.com/statistics/246888/value-of-the-global-video-game-market/, (last visited: Mar 25 2019)
Romano J, Kromrey JD, Coraggio J, Skowronek J, Devine L (2006) Exploring methods for evaluating group differences on the NSSE and other surveys: Are the t-test and Cohen's d indices the most appropriate choices. In: Annual meeting of the Southern Association for Institutional Research
Schreier J (2018) Inside Rockstar Games,' culture of crunch. https://kotaku.com/inside-rockstar-games-culture-of-crunch-1829936466, (last visited: Mar 25 2019)
Sixen (2010) How to write a good bug report. https://us.battle.net/forums/en/sc2/topic/188789092, (last visited: Mar 25 2019)
Sjöblom M, Hamari J (2017) Why do people watch others play video games? An empirical study on the motivations of Twitch users. Comput Hum Behav 75:985–996
Summerville A, Guzdial M, Mateas M, Riedl MO (2016) Learning player tailored content from observation: Platformer level generation from video traces using lstms. In: 12th Artificial Intelligence and Interactive Digital Entertainment Conference
Tantithamthavorn C, McIntosh S, Hassan AE, Matsumoto K (2017) An empirical comparison of model validation techniques for defect prediction models. IEEE Trans Softw Eng 43(1):1–18
Tantithamthavorn C, Hassan AE (2018) An experience report on defect modelling in practice: Pitfalls and challenges. In: Inproceedings of the International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP'18). ACM, pp 286–295
Valve (2018a) Steam community. https://steamcommunity.com/, (last visited: Mar 25, 2019)
Valve (2018b) Steam store. https://store.steampowered.com/, (last visited: Mar 25, 2019)
Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bullet 1(6):80–83
YouTube (2018) Auto-generated topic channels - youtube help. https://support.google.com/youtube/answer/2579942, (last visited: Mar 25 2019)
Zimmermann T, Premraj R, Sillito J, Breu S (2009) Improving bug tracking systems. In: 31st International Conference on Software Engineering-Companion Volume. IEEE, pp 247–250
Zimmermann T, Premraj R, Bettenburg N, Just S, Schroter A, Weiss C (2010) What makes a good bug report? IEEE Trans Softw Eng 36(5):618–643

Dayi Lin holds a Ph.D. in Computer Science from Queen's University, Canada. His research interests include mining software repositories and empirical software engineering, in particular, Software Engineering related aspects on PC games. His research aims at providing a better understanding of Software Engineering aspects on PC games, to help game developers produce games with better quality and player satisfaction. Contact him at dayi.lin@cs.queensu.ca. More information about Dayi is available on his website: http://lindayi.me.

Cor-Paul Bezemer is an assistant professor in the Electrical and Computer Engineering department at the University of Alberta. He heads the Analytics of Software, GAmes And Repository Data (ASGAARD) lab. Before that, he was a postdoctoral research fellow in the Software Analysis and Intelligence Lab (SAIL) at Queen's University in Kingston, Canada. His research interests cover a wide variety of software engineering and performance engineering-related topics. His work has been published at premier software engineering venues such as the TSE and EMSE journals and the ESEC-FSE, ICSME and ICPE conferences. He is one of the vice-chairs of the SPEC research group on DevOps Performance. Before moving to Canada, he studied at Delft University of Technology in the Netherlands, where he received his BSc (2007), MSc (2009) and PhD (2014) degree in Computer Science. For more information about Cor-Paul and the ASGAARD lab see: http://asgaard.ece.ualberta.ca/.

Ahmed E. Hassan is the Canada Research Chair (CRC) in Software Analytics, and the NSERC/BlackBerry Software Engineering Chair at the School of Computing at Queen's University, Canada. His research interests include mining software repositories, empirical software engineering, load testing, and log mining. Hassan received a PhD in Computer Science from the University of Waterloo. He spearheaded the creation of the Mining Software Repositories (MSR) conference and its research community. Hassan also serves on the editorial boards of IEEE Transactions on Software Engineering, Springer Journal of Empirical Software Engineering, Springer Journal of Computing, and PeerJ Computer Science. Contact ahmed@cs.queensu.ca. More information at: http://sail.cs.queensu.ca/.