Towards Using Gameplay Videos for Detecting Issues in Video Games*

Emanuela Guglielmi
STAKE Lab
University of Molise
Italy

Simone Scalabrino
STAKE Lab
University of Molise
Italy

Gabriele Bavota
SEART @ Software Institute
Università della Svizzera italiana
Switzerland

Rocco Oliveto
STAKE Lab
University of Molise
Italy

ABSTRACT

Context. The game industry is increasingly growing in recent years. Every day, millions of people play video games, not only as a hobby, but also for professional competitions (*e.g.,* e-sports or speed-running) or for making business by entertaining others (*e.g.,* streamers). The latter daily produce a large amount of gameplay videos in which they also comment live what they experience. Since no software and, thus, no video game is perfect, streamers may encounter several problems (such as bugs, glitches, or performance issues). However, it is unlikely that they explicitly report such issues to developers. The identified problems may negatively impact the user's gaming experience and, in turn, can harm the reputation of the game and of the producer. *Objective.* We aim at proposing and empirically evaluating GELID, an approach for automatically extracting relevant information from gameplay videos by (i) identifying video segments in which streamers experienced anomalies; (ii) categorizing them based on their type and context in which appear (*e.g.,* bugs or glitches appearing in a specific level or scene of the game); and (iii) clustering segments that regard the same specific issue. *Method.* We will build on top of existing approaches able to identify videos that are relevant for a specific video game. These represent the input of GELID that processes them to achieve the defined objectives. We will experiment GELID on several gameplay videos to understand the extent to which each of its steps is effective.

CCS CONCEPTS

• Software and its engineering → Software evolution; Maintaining software; *Software defect analysis.*

KEYWORDS

video games, gameplay videos, mining software repositories

ACM Reference Format:
Emanuela Guglielmi, Simone Scalabrino, Gabriele Bavota, and Rocco Oliveto. 2018. Towards Using Gameplay Videos for Detecting Issues in Video Games. In *Proceedings of MSR '22: Proceedings of the 19th International Conference on Mining Software Repositories (MSR 2022).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Video games are becoming an increasingly important form of expression in Today's culture. Their sociological, economic, and technological impact is well recognized in the literature [13] and their wide diffusion, particularly among the younger generations, has contributed to the growth of the gaming industry in several directions. Playing video games is progressively becoming a work for many: Some play for professional competitions (*e.g.,* in e-sports or speed-running), while others play to entertain others (*e.g.,* streamers) especially on dedicated platforms such as Twitch[1]. Besides all challenges that are common to software systems, developing and maintaining video games poses additional difficulties related to complex graphical user interfaces, performance requirements, and higher testing complexity. Concerning the latter point, games tend to have a large number of states that can be reached through different choices made by the player. In such a context, writing automated tests is far from trivial due to the need for an "intelligent" interaction triggering the states exploration. Even assuming such ability to explore the game space, determining what the correct behavior is in a specific state usually requires human assessment, with the exception of bugs causing the game to crash. Finally, additional complexity is brought by the non-determinism that occurs in games because of multi-threading, distributed computing, artificial intelligence and randomness injected to increase the difficulty of the game [18].

Because of the few automated approaches available for quality control in video game development [22], many games are released with unknown problems that are revealed only once customers start playing [28]. Since many streamers daily publish hours of gameplay videos, it is very likely that some of them experience such issues and leave traces of them in the uploaded videos. An example is available in [10]: The game crashes as soon as the player

* This study was accepted at the MSR 2022 Registered Reports Track.

MSR 2022, May 23–24, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

[1] https://twitch.tv

performs a specific action. The large amount of publicly available gameplay videos, therefore, might be a goldmine of information for developers. Indeed, such videos not only contain information about which kinds of issues affect a video game, but they also provide examples of interactions that led to the issue in the first place, allowing its reproduction. In their seminal work on this topic, Lin *et al.* [16] defined an approach able to automatically identify videos containing bug reports. However, such an approach mostly relies on the video metadata (*e.g.,* its length) and it is not able to pinpoint the specific parts of the video in which the bug is reported. This makes it unsuitable as a reporting tool for game developers, especially when long videos, which are not uncommon, are spot as bug-reporting.

Our goal is to introduce GELID (GamEpLay Issue Detector), an automated approach that aims at automatically extracting meaningful segments of gameplay videos in which streamers report issues and hierarchically organize them. Given some gameplay videos as input, GELID (i) partitions them in meaningful segments, (ii) automatically distinguishes informative segments from non-informative ones by also determining the type of reported issue (*e.g.,* bug, performance-related), (iii) groups them based on the "context" in which they appear (*i.e.,* whether the issue manifests itself in a specific game area), and (iv) clusters fragments related to the same specific issue (*e.g.,* the game crashes when a specific item is collected). In this registered report, we present the plan of an exploratory study to empirically evaluate GELID . We first aim to extract training data for the machine learning model we plan to use to categorize segments. To this end, we will use the approach by Lin *et al.* [16] to identify candidate videos from which we can manually label segments in which the streamer is reporting a bug. Then, we will run GELID on a set of real gameplay videos and validate its components by manually determining to what extent: (i) the extracted segments are usable, by annotating their *interpretability* (*i.e.,* they can be used as standalone videos) and *atomicity* (*i.e.,* they can not be further split); (ii) the category determined by GELID is correct, by computing typical metrics used to evaluate ML models (*e.g.,* accuracy and AUC); (iii) the clusters identified in terms of context and specific issues are valid, by performing both a quantitative (*e.g.,* using metrics such as the MoJoFM [29]) and qualitative analysis of the obtained clusters. Finally, we will validate GELID as a whole and, specifically, the usefulness of the information it provides by running a survey with developers.

2 BACKGROUND AND RELATED WORK

Several works have focused the attention on the quality assurance of video games analyzing the differences between traditional software development and video games development [18, 22]. Given the goal of our approach, we mainly focus our discussion on approaches aimed at mining and manipulating gameplay videos for different goals. Also, since GELID aims at automatically categorizing video segments, we also discuss existing taxonomies of video game issues we will use as a starting point to define our categories.

Table 1: Mapping between types of issues identified by GELID and categories from the taxonomy by Truelove ***et al.*** [28].

Issue Type	Description	Categories [28]
Logic	Issues related to the game logic, regardless of how information is presented to the player.	Object Persistence Collision of Objects Inter. btw. Obj. Prop. Position of Object Context State Crash Event Occurrence Interrupted Event Triggered Event Action Value
Presentation	Issues related to the game interface (graphical- or audio-related).	Game Graphics Information Bounds Camera Audio User Interface
Balance	Detrimental aspects in terms of "fun".	Artificial Intelligence Exploit
Performance	Performance-related issues (*e.g.,* FPS drops).	Implem. Response

2.1 Mining of Gameplay Videos

Some works targeted the automated generation of a comprehensive description of what happens in gameplay videos (*i.e.,* game commentary). Examples of these works are the framework by Guzdial *et al.* [11] and the approach presented by Li *et al.* [15] modeling the generation of commentaries as a sequence-to-sequence problem, converting video clips to commentary. On the same line of research, Shah *et al.* [24] presented an approach to generate automatic comments for videos by using deep convolutional neural networks.

The main goal of our approach, however, is to detect issues in gameplay videos. To the best of our knowledge, the only work aimed at achieving a similar goal is the one by Lin *et al.* [16]. The authors investigate whether videos in which the streamer (player) experiences faults can be automatically identified from their metadata. They observe that naïve approaches based on keywords matching are inaccurate. Therefore, they propose an approach that uses a Random Forest classifier [12] to categorize gameplay videos based on their probability of reporting a bug. Lin et al. [16] rely on Steam[2] to find videos related to specific games. While Steam is mainly a marketplace for video games, it also allows users to interact with each other and share videos. On a daily basis, for 21.4% of the games on Steam, users share 50 game videos, and a median of 13 hours of video runtime [16]. Hence, manually watching long gameplay videos classified as buggy still requires a considerable manual effort. GELID aims at reducing the effort required by developers by segmenting videos and augmenting the provided information, by including also (i) the type of issue found, (ii) the context (*i.e.,* area

[2] https://steamcommunity.com/

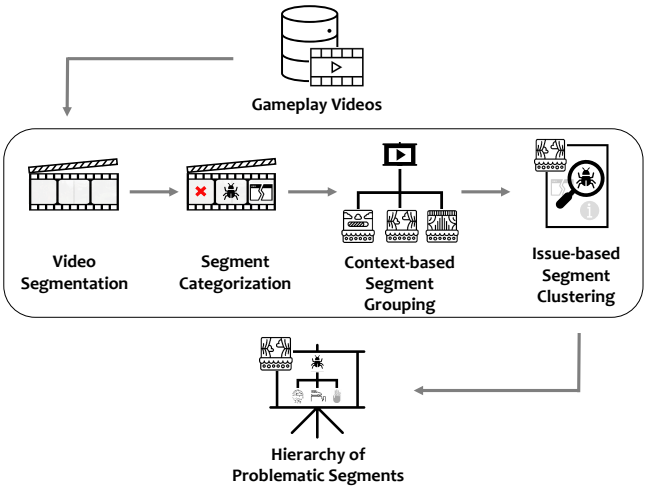

Figure 1: The workflow of GELID

of the game) in which it occurred, and (iii) other segments in which the same issue was reported (possibly from different videos).

2.2 Taxonomies of Video Game Issues

Video games can suffer from a vast variety of problems. Lin *et al.* [16] do not distinguish among the types of issues reported in the videos identified as "bug reporting", while this is one of our goals.

A recent taxonomy of issues in video games by Truelove *et al.* [28] (which extends the one by Lewis *et al.* [14]) reports 20 different kinds of issues. We will use such a taxonomy as a base to define the labels we want to assign to the video segments. However, using all such labels might be counterproductive since it is likely to observe a long-tail distribution (i.e., a few types of issues appear in most of the video fragments, while several other issues are quite rare or do not even appear). Therefore, starting from such a taxonomy, we define macro-categories by clustering similar fine-grained categories. We identified four labels, as reported in Table 1: *Logic*, *Presentation*, *Balance*, and *Performance*.

3 GELID IN A NUTSHELL

GELID takes as input a set of gameplay videos related to a specific video game, and it returns a hierarchy of segments of gameplay videos organized on three levels: (i) context (*e.g.,* level or game area), (ii) issue type (*e.g.,* bug or glitch), and (iii) specific issue (*e.g.,* game crashes when talking to a specific non-player character).

Fig. 1 shows an overview of the GELID workflow. We describe below in more detail the main steps of GELID.

3.1 Video Segmentation

The first step of GELID consists in partitioning the video in meaningful segments that can be later analyzed as standalone shorter videos. In other words, GELID aims at finding a set of "cut points" in the video. In the computer vision literature, a similar problem is referred to as "shot transitions detection". The aim is to detect sudden changes in the video content. An example of approaches defined to solve such a problem is the one introduced by Tang *et al.* [26]. Video-related information, however, might not be sufficient to find cuts in gameplay contents. For example, if the game crashes and a shot transition detection approach is used to cut the video, the second in which the crash happens would probably be selected for segmentation. The streamer, however, might need a few seconds to react to such an event by commenting what happened providing useful information for the game developers. Thus, by using shot transitions as cut points, the spoken content related to the issue might be erroneously put in the subsequent segment. To solve this problem, GELID relies on a blended video- and subtitle-based cut point detection algorithm. First, we detect shot transitions using the technique defined by Tang *et al.* [26]. Then, we shift each shot transition detected by k seconds (where k is a parameter that we will tune as part of our experiments), to account for the reaction time of the streamer. Finally, for each shifted shot transition, we will set a cut point at the moment in which the sentence pronounced by the streamer terminates, based on subtitles data. For example, let us consider the case in which a shot transition is detected at the minute 13:45 (mm:ss) with $k = 5$. GELID first shifts the detected shot transition at 13:50 (13:45 + k). Assuming that at 13:50 the streamer is in the middle of a sentence, and they finish pronouncing it at 14:05, GELID sets a cut point at 14:05.

3.2 Segment Categorization

In this second step, GELID aims at categorizing segments based on their content. GELID considers five labels: One for *non-informative* segments (*i.e.,* the ones not reporting issues), and four for *informative* segments (*i.e.,* the ones reported in Table 1). Non-informative segments are discarded and not considered in the next steps.

Previous work successfully used machine-learning to solve similar classification problems in the context of mobile app reviews [4, 23]. Such approaches mainly rely on textual features. In our context, we also have video and audio, which could help to correctly classify the segments. For example, segments with no video might be more likely to be *non-informative*, even if a comment by the player is present. Therefore, we include in GELID also video- and audio-based features. We will test different machine-learning algorithms to check which one allows to obtain the best results. Specifically, we will evaluate Random Forest [12], Logistic Regression [20], and Neural Network [8]. It is worth noting that, regardless of how precisely it is detected, a segment may show several issues at a time. For example, the game may start lagging and, at the same time, graphical glitches appear. At this stage, we do not handle segments reporting more than a issue at a time. In other words, we assume that each segment has exactly one label.

3.3 Context-based Segment Grouping

After having collected and categorized segments that contain anomalies, we group them accordingly to their *context*. With "context" we refer to the part of the game (*e.g.,* a specific game level or area) in which the anomaly occurred. This may be helpful to provide the videos to the team in charge of the development of that specific part of the game.

Such a step is important for two reasons: (i) Developers analyzing hundreds of videos related to a specific game may experience

information overload and this, in turn, would reduce the effectiveness of the video segments filtering step; (ii) Knowing the context in which more anomalies occur allows the developer to identify where attention needs to be focused to improve the gaming experience. To achieve this goal, we will rely on video information: The assumption is that videos with similar frames regard, most likely, the same context. Since the number of scenes is not necessarily known *a priori*, we will use a non-parametric clustering technique. Specifically, we will experiment with DBSCAN [7], OPTICS [1], and Mean Shift [9]. We use as the distance metric for clustering a measure based on the image similarity [19, 25]. Specifically, given two videos, A and B, we first detect the key-frames in both the videos; Then, we compute the image similarity of each key-frame of A with each key-frame of B. Finally, we compute the average.

3.4 Issue-based Segment Clustering

A set of video segments of the same kind (*e.g.,* bugs) and reported in the same context might still be hard to manually analyze for developers. For example, if 100 segments report bugs for a given level, developers need to manually analyze all of them. It might be the case, however, that most of them report the same specific bug (*e.g.,* a game object disappears). To reduce the effort required to analyze such information, we cluster segments reporting the same specific issue. This would allow developers to analyze a single segment for each cluster to have an overview of the problems affecting the specific area of the game. To achieve this goal, we will rely on both textual and image-based features, and we will use non-parametric clustering to create homogeneous groups. Textual features can help grasping the broad context (*e.g.,* objects disappearing or anomalous dialogues). Image-based features can help finding visually similar problems (*e.g.,* in the case of glitches). Similarly to the previous step, we will test several non-parametric clustering algorithms, such as DBSCAN [7], OPTICS [1], and Mean Shift [9].

4 RESEARCH QUESTIONS

The *goal* of our study is to understand to what extent GELID allows to extract meaningful information from gameplay videos.

To achieve this goal, our study is steered by the following research questions (RQs).

> RQ_1: *How meaningful are the gameplay video segments extracted by GELID?*

The first RQ aims at evaluating the quality of the segments extracted by GELID from gameplay videos in terms of their *interpretability* and *atomicity*.

> RQ_2: *To what extent is GELID able to categorize gameplay video segments?*

With this second RQ we want to understand which features and which classification algorithm allow to train the best model for categorizing gameplay video segments (second step of GELID), and what is the accuracy of such a model.

> RQ_3: *What is the effectiveness of GELID in grouping gameplay video segments by context?*

In the third RQ we aim to define the best clustering algorithm for grouping segments based on the game context (third step of GELID), and how effective is such an algotithm in absolute terms.

> RQ_4: *What is the effectiveness of GELID in clustering gameplay video segments based on the specific issue?*

Similarly to RQ_3, with this last RQ, we want to understand which features and clustering algorithm allow to achieve the best results for clustering segments based on the specific issue (fourth step of GELID), and how effective is such an algorithm in absolute terms.

> RQ_5: *To what extent is the information provided by GELID useful to practitioners?*

With this last RQ, we want to evaluate GELID as a whole. Specifically, we want to understand the perceived usefulness of the information provided by GELID, and which pieces of information are more relevant than others according to practitioners.

5 DATASETS

To answer our RQs and validate the defined approach, we will rely on gameplay videos from YouTube. While other platforms, even more video game-oriented, could be used (*e.g.,* Twitch), YouTube provides APIs for searching videos of interest and it also allows to download videos including subtitles, which are required by GELID. While subtitles can be automatically generated when the video lacks them, the results could be noisy and, in this phase, we aim at evaluating GELID assuming high-quality input data.

In our study, we will collect three datasets. The first one, composed by video segments, will be used for training the supervised model used in step 2 of GELID (*i.e.,* segment categorization). The second one, composed by complete videos, will be used for evaluating the single components of GELID and answer RQ_{1-4}. The third one, composed by the output of GELID on a set of gameplay videos of a specific video game, will be used for evaluating GELID as a whole with practitioners. We will publicly release all the datasets to foster future research in this field.

5.1 Training Data

We will use the APIs provided by YouTube to select a random sample of gameplay videos in English from a diverse set of video games. While our premise is that several gameplay videos report issues, we also expect that the issues-reporting videos represent a minority of the entire gameplay videos population (thus the relevance of our approach). Therefore, to support the construction of the dataset containing training data for the categorization step, we will use the approach defined by Lin *et al.* [16] and consider only videos identified as issue-reporting. Since our approach works at segment-level, we expect to collect a sufficient number of non-informative segments from videos reporting issues. Thus, the issue-reporting videos will provide training data for all categories of segments (*i.e.,* non-informative plus the four informative sub-categories). Also,

the approach we will use is not perfect, so we will end up analyzing videos not reporting bugs anyway. At this stage, we will only include videos with subtitles since GELID relies on NLP-based features computed on them. Some YouTube videos have manually-defined subtitles, while others have automatically generated ones. We include both of them. Indeed, while it is possible that the second category contain errors, this risk also exists in manually generated ones. Also, the general quality of the subtitles generated by YouTube is generally quite high for the English language.

A human evaluator will manually split each video into meaningful segments, and at least two human evaluators will label each segment as logic, presentation, balance, performance, or non-informative. At this stage, our goal is to collect at least 1,000 labeled segments. To achieve this goal, we will run the previously mentioned process on batches of 500 videos at a time, until we collect the desired target number of segments. For each batch, we will randomly sample YouTube videos matching a generic query (*e.g.,* "gameplay") to ensure that the final sample is representative enough. We choose to include at least 1,000 segments in the training set because (i) such a number would probably allow us to appropriately train the model, and (ii) it is sufficient to have a representative set of segments. Indeed, assuming an infinite population (*i.e.,* we have an indefinitely high number of segments) and a 95% confidence level, a sample of 1,000 segments allows us to have a 3.1% margin of error, which we find acceptable in this context. To understand whether it is feasible to collect such an number of segments, we run a preliminary analysis. We searched for YouTube videos regarding the popular game Grand Theft Auto 5 (GTA 5). We found a total of 9,460,000 gameplay videos results. Given such a high number of results for a single video game, we believe that finding 1,000 segments is feasible.

Finally, it is possible that the training set contains a few instances for some categories of issue types. For this purpose, if the training set is unbalanced, we will use oversampling techniques (*e.g.,* SMOTE [3]) to generate synthetic instances for underrepresented categories.

5.2 Components Validation Data

To select videos on which we will validate the single components of GELID, we will first need to select a small set of specific video games. Indeed, the third and fourth steps of GELID are reasonable only when segments from the same video game are given. To select the video games to use, we will rely on the information available on Steam, one of the largest video game marketplaces [27]. We will select three video games that are both popular (*e.g.,* for which many gameplay videos exist) and that had several reported issues (*e.g.,* for which GELID gives the best advantage). More specifically, we will select video games with many downloads, low review scores and many patches. Then, we will collect a random sample of gameplay videos related to each selected video game from YouTube. Also in this case, we will select only English videos with subtitles (either manually added or automatically generated). Since we will test different machine-learning techniques (both for categorization and clustering), we will need to tune their respective hyper-parameters. To this aim, we will use 10% of the data acquired at this stage as *evaluation set*, and the remaining 90% as test set.

Table 2: Questions for the survey we will run to answer RQ_5.

	Question	Type of response
Pre-quest.	Full name	Text
	Email address	Text
	Education	Multiple selection (*e.g.,* graduate)
	Role	Multiple selection (*e.g.,* tester)
	Years of experience	Number
Questionnaire	**Context information**	
	To what extent is context summary information useful?	5-point Likert scale
	To what extent is the ability to navigate segments from contexts useful?	5-point Likert scale
	Please justify your answers	Open response
	Issue category information	
	To what extent is issue category summary information useful?	5-point Likert scale
	To what extent is the ability to navigate segments from issue categories useful?	5-point Likert scale
	Please justify your answers	Open response
	Speci□c issue information	
	To what extent is specific issue summary information useful?	5-point Likert scale
	To what extent is the ability to navigate segments from clusters of specific issues useful?	5-point Likert scale
	Please justify your answers	Open response
	Segments information	
	How useful would the segments be to understand issues?	5-point Likert scale
	How useful would the segments be to reproduce issues?	5-point Likert scale
	Global evaluation	
	How useful would GELID be during the closed testing phase?	5-point Likert scale
	How useful would GELID be during the beta testing phase?	5-point Likert scale
	How useful would GELID be during the production phase?	5-point Likert scale
	What are the strengths of GELID?	Open response
	What are the weaknesses of GELID?	Open response
	Additional comments (optional)	Open response

5.3 Approach Validation Data

To select videos on which we will validate GELID as a whole, we will first select a video game on Steam and extract a set of gameplay videos from YouTube about such a game. To do this, we will use the exact same approach used to build the previously described dataset, but with a fourth game. Then, we will feed GELID with such videos. GELID, in turn, will provide information about (i) contexts (*i.e.,* area of the game), (ii) issue types (*e.g.,* logic or presentation issue), and (iii) specific issue. At this stage, we will use the best configuration of GELID, based on the findings of RQ_{1-4} (*e.g.,* the most accurate categorization model). We will ask practitioners to evaluate the information provided by GELID, as detailed in Section 6.

6 EXECUTION PLAN

We will run the following plan to answer our research questions and conduct the study. We summarize the execution plan in Fig. 2.

6.1 Research Method for RQ_1

To answer RQ_1, we will evaluate the technique we defined with different values of k (streamer reaction times). Specifically, we will instantiate our approach with k in the set $\{0, 5, 10\}$ seconds.

We will evaluate the segments detected by each variant of our approach in terms of their (i) *interpretability* (*i.e.,* it is possible to watch the segment and acquire all the information needed to understand what has been experienced by the streamer) (ii) the *atomicity* (*i.e.,* it is not possible to further split the segments). Such aspects are complementary: It would be possible to maximize the *interpretability* by creating few segments (*e.g.,* just one for the whole video); this, however, would result in lower *atomicity* since the segments could be further divided into parts. Two human annotators will watch segments generated by each technique and manually annotate each

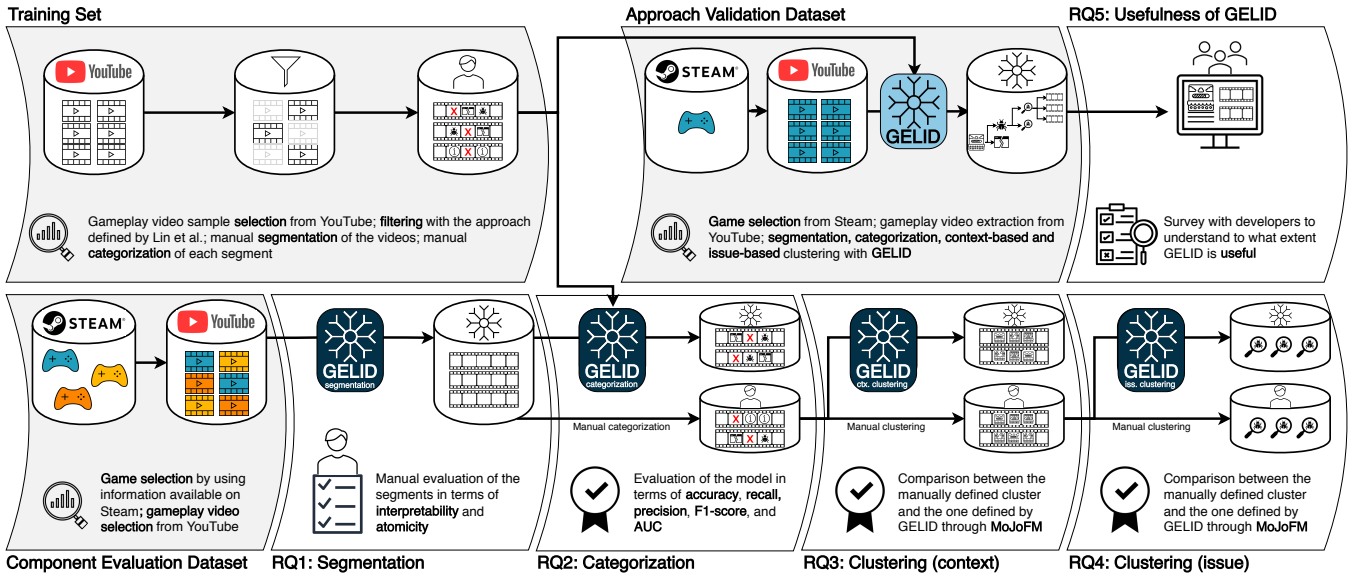

Figure 2: Summary of the study design.

segment in terms of its *interpretability* and *atomicity* on a 5-point Likert scale. As for the first metric, we will ask the annotator to evaluate to what extent he/she can fully understand what is happening based only on the segment itself. As for atomicity, instead, we will ask annotators to assess if the segment can be further divided in additional standalone (fully interpretable) segments. The final score should be computed as 5 minus the number of additional standalone segments that can be further extracted, or 1 if more than four standalone segments are found. We will report the inter-rater reliability between the annotators by using the Cohen's kappa coefficient [6]. Then, for each segment, we will compute the mean and median *interpretability* and *atomicity*. Finally, we will compare the tested techniques in terms of such metrics using a Mann-Whitney U test [17], and adjusting the p-values resulting for multiple comparisons using the Benjamini and Hochberg procedure [2]. We will also report the effect size, using the Cliff's delta [5], to understand the magnitude of differences observed. We run power analysis to define an adequate sample size that allows us to detect possible differences in terms of interpretability and atomicity among the three groups of segments (*i.e.*, with $k \in \{0, 5, 10\}$). First, we are interested in observing variations of at least 0.5 in the means of the groups for both the metrics; smaller variations would likely be practically irrelevant. To determine the expected standard deviation, we run 1,000 simulations in which we randomly assigned scores between 1 and 5, and we computed, for each simulated assignment, the standard deviation. We obtained values in the range $[1.28, 1.54]$. If we analyze 200 segments for each group, assuming the previously reported range of expected standard deviation values, we expect to achieve a power in the range $[90\%, 97\%]$. If we simulate the worst-case scenario in terms of standard deviation (*i.e.*, we have half observations with the lowest score and the other half with the highest score), we obtain a standard deviation of ~ 2, which leads to a 71% power. Even in such an unlikely scenario, we would have acceptable power. Therefore, we plan to use groups of 200 segments.

6.2 Research Method for RQ_2

To answer RQ_2, we will train and test several approaches for categorization, using different sets of features and different machine learning algorithms to understand (i) which features are more relevant, and (ii) which machine learning technique allows to achieve the best results. For each tested algorithm (*i.e.*, Random Forest [12], Logistic Regression [20], Neural Network [8]), we will define three models including (i) textual features only, extracted from the subtitles (*i.e.*, what the streamer says), (ii) video features only, extracted from the video (*i.e.*, what happens in the game), and (iii) all features together. In total, we will compare nine models. We will explore different types of NLP features, including bag-of-words [30] and word2vec [21].

To train each model, we will use the training set described in Section 5. To define a *test set*, instead, we will consider the segments obtained through the best segmentation technique, based on the results of RQ_1. Two human annotators will independently analyze each segment and manually label them as logic, presentation, balance, performance, or non-informative. The human annotators will discuss the cases of disagreement aiming at reaching consensus. If no consensus can be found, the segment will be discarded as ambiguous also for human evaluators.

For each model, we will report the global accuracy (*i.e.*, percentage of correctly classified instances) and, for each class, the achieved precision ($\frac{TP}{TP+FP}$), recall ($\frac{TP}{TP+FN}$), and AUC (Area Under the ROC Curve).

6.3 Research Method for RQ_3 and RQ_4

To answer both RQ_3 and RQ_4, we will test the following non-parametric clustering techniques: DBSCAN [7], OPTICS [1], and

Mean Shift [9]. We will start from the test set defined to answer RQ_2. For each video game, two human annotators will group the segments based on the game context in which they appear. The human annotators will discuss conflicts aiming at reaching consensus. Segments on which consensus can not be reached will be discarded, similarly to RQ_2. We will use the ground-truth partition produced after this step to evaluate the previously-mentioned clustering techniques to answer RQ_3 (more on this below). Then, for each cluster and for each category of issues, we will further cluster segments according to the specific issue highlighted. We will use the same process used for building the ground-truth partition defined for answering RQ_3, and we will use such a partition to answer RQ_4.

For both the RQs, we will compare the models by using the MoJo eFfectiveness Measure (*MoJoFM*) [29], a normalized variant of the MoJo distance. *MoJoFM* is computed using the following formula:

$$MoJoFM(A,B) = 100 - (\frac{mno(A,B)}{max(mno(\forall E_A,B))} \times 100)$$

where $mno(A,B)$ is the minimum number of *Move* or *Join* operations one needs to perform in order to transform a partition A into a different partition B, and $max(mno(\forall E_A,B)$ is the maximum possible distance of any partition A from any partition B. *MoJoFM* returns 0 if partition A is the farthest partition away from B; it returns 100 if A is equal to B.

6.4 Research Method for RQ_5

To answer RQ_5, we will run a survey with at least five practitioners, aimed at collecting their opinions on the usefulness of the information provided by GELID. We preliminarly acquired the availability of five professional developers to achieve do this. The survey will be composed of three steps. First, participants will complete a pre-questionnaire to acquire basic information. We report the specific questions we will ask in the top part of Table 2. Second, we will ask them to freely browse for 15 minutes a web-app containing the information generated by GELID for a specific video game. To this aim, we will exploit the third dataset defined in Section 5. Especially, participants will be able to: (i) view summary information about the contexts; (ii) browse contexts and view summary information about the categories of issues affecting them; (iii) browse issue categories and view summary information about clusters of video segments regarding specific issues; (iv) browse such clusters and watch video segments. After this step, participants will answer specific questions about the perceived usefulness of GELID, as specified in the bottom part of Table 2. We will report summary statistics about the responses and qualitatively analyze their comments to also provide insights about future research directions.

7 LIMITATIONS, CHALLENGES, AND MITIGATIONS

In this step we summarize the main limitations of our work and outline the mitigation strategies we use.

Subjectivity of the Manual Analysis. It is possible that the manually determined labels and partitions used to answer our RQs are not correct. To mitigate this limitation, all the manual evaluations are performed by two authors, and the results are discussed to reach consensus.

Evaluation Biases. In the evaluation of GELID, we will explicitly select video games with many issues. This could result in a bias in the evaluation. It might be possible that we conclude that GELID works, while it works only on problematic games. However, we believe that GELID is most useful for such a category of video games

Incomplete De□nition of Categorization Labels. The definition of the labels used in the second step of GELID (*i.e.,* categorization) might be incomplete. It is possible that we do not consider some relevant categories of issues. To mitigate this limitation, we relied on a state-of-the-art taxonomy [28].

Ine□ectiveness of the Features. A key issue in implementing GELID consists in defining meaningful features for categorizing and clustering video segments. We will initially rely on features previously engineered in a similar context (*e.g.,* categorization and clustering of mobile app reviews [4, 23]). A different set of features may lead to different, possibly better, results.

Manual Analysis E□ort. We foresee a big amount of manual analysis to be performed to both (i) build a training set and (ii) answer our RQs. To address this challenge, we will likely involve other people in this process (*e.g.,* game players).

8 CONCLUSION

In recent years, there has been a growing interest in video games. During game development, many bugs go undetected prior to release because of the difficulty of fully testing all aspects of a video game. We introduce GELID, a novel approach for detecting anomalies in video games from gameplay videos to support developers by providing them with useful information on how to improve their games. We presented the plan of our empirical study designed to understand to what extent GELID can provide meaningful information to developers.

REFERENCES

[1] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod record* 28, 2 (1999), 49–60.

[2] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* 57, 1 (1995), 289–300.

[3] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[4] Ning Chen, Jialiu Lin, Steven CH Hoi, Xiaokui Xiao, and Boshen Zhang. 2014. AR-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th international conference on software engineering*. 767–778.

[5] Norman Cliff. 1993. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological bulletin* 114, 3 (1993), 494.

[6] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20, 1 (1960), 37–46.

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *kdd*, Vol. 96. 226–231.

[8] James Franklin. 2005. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer* 27, 2 (2005), 83–85.

[9] Keinosuke Fukunaga and Larry Hostetler. 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory* 21, 1 (1975), 32–40.

[10] The WP Guru. 2020. Cyberjunk 2077: Corpo Crash. https://youtu.be/ybvXzSLy9Ew?t=1448.

[11] Matthew Guzdial, Shukan Shah, and Mark Riedl. 2018. Towards Automated Let's Play Commentary. *arXiv preprint arXiv:1809.09424* (2018).

[12] Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, Vol. 1. IEEE, 278–282.

[13] Steven E Jones. 2008. *The meaning of video games: Gaming and textual strategies*. Routledge.
[14] Chris Lewis, Jim Whitehead, and Noah Wardrip-Fruin. 2010. What went wrong: a taxonomy of video game bugs. In *Proceedings of the fifth international conference on the foundations of digital games*. 108–115.
[15] Chengxi Li, Sagar Gandhi, and Brent Harrison. 2019. End-to-end let's play commentary generation using multi-modal video representations. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*. 1–7.
[16] Dayi Lin, Cor-Paul Bezemer, and Ahmed E Hassan. 2019. Identifying gameplay videos that exhibit bugs in computer games. *Empirical Software Engineering* 24, 6 (2019), 4006–4033.
[17] Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics* (1947), 50–60.
[18] Emerson Murphy-Hill, Thomas Zimmermann, and Nachiappan Nagappan. 2014. Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development?. In *Proceedings of the 36th International Conference on Software Engineering*. 1–11.
[19] Silvia M Ojeda, Pedro W Lamberti, and Ronny O Vallejos. 2012. Measure of similarity between images based on the codispersion coefficient. *Journal of Electronic Imaging* 21, 2 (2012), 023019.
[20] M Revan Özkale, Stanley Lemeshow, and Rodney Sturdivant. 2018. Logistic regression diagnostics in ridge regression. *Computational Statistics* 33, 2 (2018), 563–593.
[21] Xin Rong. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738* (2014).
[22] Ronnie ES Santos, Cleyton VC Magalhães, Luiz Fernando Capretz, Jorge S Correia-Neto, Fabio QB da Silva, and Abdelrahman Saher. 2018. Computer games are serious business and so is their quality: particularities of software testing in game development from the perspective of practitioners. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 1–10.
[23] Simone Scalabrino, Gabriele Bavota, Barbara Russo, Massimiliano Di Penta, and Rocco Oliveto. 2017. Listening to the crowd for the release planning of mobile apps. *IEEE Transactions on Software Engineering* 45, 1 (2017), 68–86.
[24] Shukan Shah, Matthew Guzdial, and Mark O Riedl. 2019. Automated Let's Play Commentary. *arXiv preprint arXiv:1909.02195* (2019).
[25] Eli Shechtman and Michal Irani. 2007. Matching local self-similarities across images and videos. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.
[26] Shitao Tang, Litong Feng, Zhanghui Kuang, Yimin Chen, and Wei Zhang. 2018. Fast video shot transition localization with deep structured models. In *Asian Conference on Computer Vision*. Springer, 577–592.
[27] Eric J Toy, Jaya VHH Kummaragunta, and Jin Soung Yoo. 2018. Large-scale cross-country analysis of steam popularity. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 1054–1058.
[28] Andrew Truelove, Eduardo Santana de Almeida, and Iftekhar Ahmed. 2021. 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 (ICSE)*. IEEE, 736–747.
[29] Zhihua Wen and Vassilios Tzerpos. 2004. An effectiveness measure for software clustering algorithms. In *Proceedings. 12th IEEE International Workshop on Program Comprehension, 2004.* IEEE, 194–203.
[30] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics* 1, 1 (2010), 43–52.