A Replication Package for It Takes Two to TANGO: Combining Visual and Textual Information for Detecting Duplicate Video-Based Bug Reports

Nathan Cooper*, Carlos Bernal-Cárdenas*, Oscar Chaparro*, Kevin Moran†, Denys Poshyvanyk*
*College of William & Mary (Williamsburg, VA, USA), †George Mason University (Fairfax, VA, USA)
nacooper01@email.wm.edu, cebernal@cs.wm.edu, oscarch@wm.edu, kpmoran@gmu.edu, denys@cs.wm.edu

***Abstract*—When a bug manifests in a user-facing application, it is likely to be exposed through the graphical user interface (GUI). Given the importance of visual information to the process of identifying and understanding such bugs, users are increasingly making use of screenshots and screen-recordings as a means to report issues to developers. Due to their graphical nature, screen-recordings present challenges for automated analysis that preclude the use of current duplicate bug report detection techniques. This paper describes in detail our reproduction package artifact for TANGO, a duplicate detection technique that operates purely on video-based bug reports by leveraging *both* visual and textual information to overcome these challenges and aid developers in this task. Specifically, this reproduction package contains the data and code that enables our TANGO's empirical evaluation replication and future research in the area of duplicate video-based bug report detection.**

***Index Terms*—Bug Reporting, Screen Recordings, Duplicate Detection**

I. INTRODUCTION

Many modern mobile applications (apps) allow users to report bugs in a graphical form, given the GUI-based nature of mobile apps. For instance, Android and iOS apps can include built-in screen-recording capabilities in order to simplify the reporting of bugs by end-users and crowd-testers [4, 6, 11]. The reporting of visual data is also supported by many crowd-testing and bug reporting services for mobile apps [2–8, 10–12], which intend to aid developers in collecting, processing, and understanding the reported bugs [18, 22].

The proliferation of sharing images to convey additional context for understanding bugs, *e.g.,* in Stack Overflow Q&As, has been steadily increasing over the last few years [23]. Given this and the increased integration of screen capture technology into mobile apps, developers are likely to face a growing set of challenges related to processing and managing app screen-recordings in order to triage and resolve bugs — and hence maintain the quality of their apps.

To aid developers in determining whether video-based bug reports depict the same bug, we developed TANGO [20], a novel approach that analyzes both visual and textual information present in mobile screen-recordings using tailored computer vision (CV) and text retrieval (TR) techniques, with the goal of generating a list of candidate videos (from an issue tracker) similar to a target video-based report.

We conducted an empirical study to measure TANGO's effectiveness and ability to save developer effort in identifying duplicate video-based bug reports. The evaluation was carried out using 180 video-bug reports from six Android apps, and 4,860 duplicate detection tasks. The evaluation revealed that TANGO is able to suggest correct duplicate reports in the top-2 of the ranked candidates for 83% of duplicate detection tasks. Additionally, TANGO can reduce the time they spend in finding duplicate video-based bug reports by $\approx 65\%$.

This paper describes the artifact of TANGO's empirical evaluation. The artifact enables (i) the replication of the evaluation results as reported in our original paper [20], and (ii) future research on video-based duplicate detection, bug replication, and mobile app testing. The artifact contains 180 video-based bug reports with duplicates for six Android apps, TANGO's source code, TANGO's trained models, the defined duplicate detection tasks, TANGO's (intermediate) output, and detailed evaluation results. Additionally, the artifact includes a command-line tool that allows a user or researcher to run TANGO on additional data. The artifact has been archived for future use on Zenodo [16] and GitHub [17].

II. ARTIFACT DESCRIPTION

The artifact has four parts: (i) the source code for TANGO and its evaluation, (ii) the data used to evaluate TANGO along with the trained models, (iii) the intermediate outputs produce by TANGO and the results of TANGO's evaluation, and (iv) the command line tool for easily running TANGO's visual model on additional data.

A. Source code

We built TANGO in Python and Java. We designed it to be a tool that researchers and developers could easily use and extend, so all of the code is well documented and contains usage examples. The Python code is done via Jupyter Notebooks [14] using the nbdev library [13], which allows for TANGO's code to reside next to its documentation.

TANGO's python code has six modules, each responsible for a different API.

1) *prep*: contains functionality for data loading and preprocessing.
2) *features*: contains visual feature extractors, *i.e.,* SimCLR and SIFT.
3) *eval*: contains functionality and metrics for calculating the performance of TANGO.
4) *model*: contains implementation for SimCLR along with the similarity functions for the types of TANGO, *i.e.,* longest common subsequence (LCS) and weighted LCS version
5) *approach*: contains the visual versions of TANGO.
6) *combinations*: contains functionality for combining the visual and textual components together.

For easy installation we created a PyPi package[1] as well as a Docker image[2]. The Java code is stored in our data artifact and is discussed in the following section. The artifact contains a README file outlining the steps to install and replicate the results of TANGO's evaluation. We release TANGO's source code under the Apache v2.0 licence.

B. Data and trained models

The data directory [16] contains the video-based bug reports for six open source apps that we collected in a user study (see our original paper for more details [20]) along with the trained models and is linked in the artifact's README. Each user created a video-based bug report for a set of bugs across different Android apps. The directory where the videos-based bug reports are located is broken down into sub-directories, each one representing a user. Within each user sub-directory, there are multiple directories representing Android apps. Each of these app directories contain sub-directories that denote bugs, and each of the bug directories contains a video-based bug report as an mp4 file.

The models directory contains the three TANGO models that we evaluated (SIFT [21], SimCLR [19], and OCR+IR [1, 9]). Each directory contains the corresponding trained codebook files that we generated for SIFT and SimCLR. These codebook files are pickle files[3] that contain the binary representation of the trained codebooks. Additionally, in the SimCLR folder, you will find a checkpoint and Pytorch [15] model file for reloading the trained SimCLR model. The OCR+IR folder contains the Java code for the OCR+IR model as well as the intermediate output.

We release this artifact's data under "Creative Commons Attribution 4.0 International" license.

C. Intermediate outputs and results

The outputs folder contains all the intermediate outputs of our code, except for OCR+IR. The results folder contains the raw rankings and metrics for the SIFT and SimCLR models for all combinations of video-based bug reports per app. The *evaluation_setting* directory contains a json file that defines the duplicate detection tasks that we used for evaluating our models, *i.e.,* setting #2 (see paper for more details[20]). The *user_rankings_weighted_all* and *user_results_weighted_all* directories contain the converted version of the raw rankings and metrics for the SIFT and SimCLR models to match setting #2. The *extracted_text* directory contains the output OCR+IR model, specifically, the frames of the videos and the text from each frame. Lastly, the *combined* directory contains the results of the combined tango approach.

D. Command-line tool

We also provide a command-line tool for running TANGO's visual model on additional data. Currently only the visual part of TANGO with SimCLR is supported and not the combined version since this version needs to be tuned depending on vocabulary overlap in an app (see original paper for more details[20]). This tool allows for a user to specify paths to a query video and a directory containing a corpus of videos that will be compared against the query video for duplicate detection. The output of this tool is a list of the similarity values (in descending order) between the query video and each video in the given corpus.

[1]https://pypi.org/project/two-to-tango/

[2]https://hub.docker.com/repository/docker/semerulab/tools/general

[3]https://docs.python.org/3/library/pickle.html

ACKNOWLEDGEMENTS

This research was supported in part by the NSF CCF-1955853 and CCF-1815186 grants. Any opinions, findings, and conclusions expressed herein are the authors' and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Tesseract ocr library https://github.com/tesseract-ocr/tesseract/wiki.
[2] Bird eats bugs https://birdeatsbug.com/, 2020.
[3] Bug squasher https://thebugsquasher.com/, 2020.
[4] Bugclipper http://bugclipper.com, 2020.
[5] Bugreplay https://www.bugreplay.com/, 2020.
[6] Bugsee https://www.bugsee.com/, 2020.
[7] Instabug https://instabug.com/screen-recording, 2020.
[8] Outklip https://outklip.com/, 2020.
[9] Python tesseract https://github.com/madmaze/pytesseract, 2020.
[10] Snaffu https://snaffu.squarespace.com/, 2020.
[11] Testfairy https://testfairy.com, 2020.
[12] Ubertesters https://ubertesters.com/bug-reporting-tools/, 2020.
[13] nbdev https://nbdev.fast.ai/, 2021.
[14] Project jupyter https://jupyter.org/, 2021.
[15] Pytorch https://pytorch.org/, 2021.
[16] Tango's data repository https://doi.org/10.5281/zenodo.4453765, 2021.
[17] Tango's github repository https://github.com/ncoop57/tango, 2021.
[18] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? In *FSE'08*, pages 308–318, New York, NY, USA, 2008. ACM.
[19] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *ICML'20*, pages 1597–1607, 2020.
[20] N. Cooper, C. Bernal-Cárdenas, O. Chaparro, K. Moran, and D. Poshyvanyk. It takes two to tango: Combining visual and textual information for detecting duplicate video-based bug reports. In *ICSE'21*, 2021.
[21] D. Lowe. Distinctive image features from scale-invariant keypoints. *JCV'04*, 60:91–, 11 2004.
[22] K. Mao, M. Harman, and Y. Jia. Crowd intelligence enhances automated mobile testing. In *ASE'17*, pages 16–26, Piscataway, NJ, USA, 2017. IEEE Press.
[23] M. Nayebi. Eye of the mind: Image processing for social coding. In *ICSE'20*, page 49–52, 2020.