Detlef Krömker und Ulrik Schroeder (Hrsg.): Die 16. E-Learning Fachtagung Informatik,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 231

Track every move of your students: log files for Learning Analytics from mobile screen recordings

Philipp Krieter[1] and Andreas Breiter[1]

Abstract: One of the main data sources for Learning Analytics are Learning Management Systems (LMS). These log files are limited though to interactions within the LMS and cannot take into account interactions of students in other applications and software in a digital learning environment. In this paper, we present an approach for generating log files based on mobile screen recordings as a data source for Learning Analytics. Logging mobile application usage is limited to rather general system events unless you have access to the source code of the operating system or applications. To address this we generate log files from mobile screen recordings by applying computer vision and machine learning methods to detect individually defined events. In closing, we discuss how these log files can be used as a data source for Learning Analytics and relevant ethical concerns.

Keywords: Learning Analytics, mobile screen recordings, data sources, log files, Human Computer Interaction, Computer Vison

1 Introduction

The field of Learning Analytics uses a wide range of data sources. A common data source are log files from systems that support the learning process like Learning Management systems [PE14]. Pardo and Kloos [PK11] state that early Learning Analytics research is ñLMS-centricò: using only data from an LMS might limit research to a small part of the activities of students. Especially communication often happens through existing email or mobile chat applications and not through the communication features of the used LMS [PK11]. This reduces the informative value of log files from LMSs. To provide a comprehensive picture and analysis of learning activities, additional data is helpful beyond the log files of the LMS.

In order to analyze learning outside the LMS, additional data is necessary, especially when various third-party applications and software are used in a digital learning environment. In these applications, it is difficult to gather data about the interactions of students. To get log data from these third-party applications, it is usually necessary to implement log commands in their source code. However, many applications are not open source and changing the source code to generate data for Learning Analytics is not possible.

[1] University of Bremen, Institute for Information Management Bremen (ifib), Am Fallturm 1, 28359 Bremen, {pkrieter|abreiter}@ifib.de

Papamitsiou and Economides state that ñ[..]every óclickô within an electronic learning environment may be valuable actual information that can be tracked and analyzed. Every simple or more complex action within such environments can be isolated, identified and classified through computational methods into meaningful patterns.ò [PE14].

Hepp et al. describe these small pieces of data we leave behind when we act in digital environments ñdigital tracesò and stress that these traces are meaningless until we put them into a relevant context by appropriate methods and their triangulation [HBF18]. While we leave more traces in digital learning environments than ever before, it is still a challenge to collect these data points in an appropriate way and give meaning to this data through adequate Learning Analytics, for example.

Our work addresses this lack of data sources in digital learning environments containing not just an LMS but also multiple closed source code applications by presenting a method to generate log files for Learning Analytics based on mobile screen recordings.

A graphical user interface (GUI) always represents the current state of a system, based on user and system events [AKRR14]. The focus on GUIs in Human Computer Interaction (HCI) leads to the point that the analysis of screen recordings makes it possible to understand almost every interaction, every behavior or every task that the user performs while using a computer system. Interpreting screen recordings manually is very time consuming, what limits this approach to relatively short time periods. Our approach utilizes computer vision and machine-learning methods to automatically analyze screen recordings and to create event-based log files. Our intention is to combine the advantages of short-term detailed screen recording analysis with long-term log file generation.

The main contribution we aim at in this paper is to introduce a new method to collect data in digital learning environments. We present how we define events of interest and find these in mobile screen recordings in order to create log files for further analysis. The results and evaluation in this paper are based on 118 example screen recordings from a mobile device. Besides that, we show basic descriptive visualizations of example data and show how this could be used as a new data source for Learning Analytics.

2 Related Work

2.1 Data Sources for Learning Analytics outside the LMS

A systematic literature search by Papamitsiou and Economides [PE14] identifies various data sources used in the field of Learning Analytics and educational data mining, such as log files of goal-oriented implemented systems, questionnaires, interviews, web tracking software, open data sets and virtual machines. A main source for data in the field of Learning Analytics are still log files containing entries about interactions of students within an LMS [TRB15, PK11].

There are several efforts in the Learning Analytics community to collect data on student behavior and interactions outside the LMS. Pardo and Kloos [PK11] present an approach using virtual machines to log a number of events outside of an LMS. They provide pre-configured virtual machines to students of a programming class that the students run on their personal computers. The machines monitor a couple of events like power-up and shutdown, start and use of selected tools and commands and browser history. They show that a significant amount of relevant interactions happen outside of the sphere of the used LMS.

Another attempt to gather data beyond the LMS is introduced by Kitto et al. [Ki15] utilizing data from social media combined with LMS data. They present an open source toolkit accessing six social media APIs and save in a standardized actor-verb-object notation for further analysis. Their focus is to gather data from multiple sources in a uniform way at large-scale, but also concerns regarding related privacy and data ownership issues.

Another study by Tempelaar et al. explores and compares the predictive power of different Learning Analytics data sources [TRG15]. Therefore, they collect self-reported data, LMS data and e-tutorial data of formative assessments. In the conducted study, data from 922 mathematics and statistics students is collected. They mention that the use of only LMS data does not have substantial predictive power in their study. They find that the use of formative computer-assisted assessments is a good predictor for detecting underperforming students.

There is still no agreement within the Learning Analytics community as to which interactions of students within a digital learning environment are decisive for effective learning [Ag14]. More research and exploration of different data sources is needed, to address this. Coming from this point, our approach, which we present in this paper, follows the path of opening up and developing new data sources with the aim of providing a more comprehensive picture and a new perspective on learning activities.

2.2 Detecting events in screen recordings

The analysis of screen recordings of student devices is not yet used in the context of Learning Analytics, neither for manual nor for automatic analysis. There are some research approaches that focus on the automatic analysis of screen captures, but not in association with the generation of data for Learning Analytics. Most research in this area is aiming at analyzing Human Computer interaction, but there are also approaches that could be used as a data source in other contexts such as learning analysis.

The project òInspectorWidgetò [Fr16] proposes an automatic screencast annotating system for usability checks using computer vision and machine learning techniques. The approach is designed for usability checks and requirement engineering scenarios. The work is in an early stage, but has a potential for long-term research studies and generating log files for

several purposes on desktop devices. The project has been developed for Linux, Windows, and Mac, but lacks to support mobile devices like smartphones and tablet computers.

An approach by Chang et al. [CYM10] has some close relation to our approach although their aim is not to generate log files from screen recordings. The focus of their project ñSikuli Testò is to test desktop GUIs automatically using computer vision methods. The script acts òlike a robotic testerò and acts on the visual screen output of desktop GUIs based on previously defined events. Common with our approach is to detect individually defined GUI states using computer vision methods.

Matejka et al. introduce a tool for the collection of data about software usage, with effort spent in making the approach independent of the actual application. This aim is closely related to our approach, but the way of data collection is different. The result of this is a dynamic heat map overlay, that shows usage patterns of the active user and of the community. They collect data about how frequently users use functions in office applications aiming at optimizing software. The tool is only available for Mircosoft Windows, using its accessibility APIs (not available for all applications), making the method useable for many Windows applications, but requires additional work for every application [MGF13].

Based on the previous work on data sources in Learning analytics and we present how we combine computer vision and machine-learning methods to automatically detect events in mobile screen recordings to generate log files for Learning Analytics.

3 Technical Approach

The process for generating log data and further analysis in form of Learning Analytics involves several steps (see Fig. 1). At first, video data needs to be collected on the studentôs devices, followed by the definition of events of interest and automatic analysis of the video material based on these event definitions. The resulting log files need to be prepared for further analysis, to be finally analyzed and put into context.

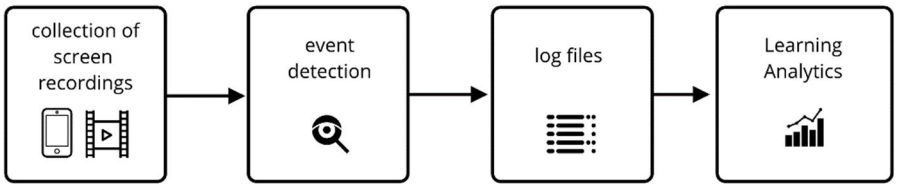

Fig. 1: Simplified process: from screen recordings to log files for Learning Analytics

3.1 Collection of screen recordings

To validate the results of our approach we recorded 118 smartphone usage sessions (68 minutes), which is roughly equivalent to the average use of a smartphone during one day [Bö11]. With this amount of video, it is still possible to manually validate the log results and identify errors in the detection process. We define a phone usage session as from switching on the screen until the screen turns off again and record interactions in the locked state as well as in unlocked device state. For evaluation of our approach, we recorded high-quality video in full HD at an average bitrate of 4418 Kbit/s and 20 frames per second (FPS) on an Android device.

3.2 Event detection

Since video analysis is a compute-intensive process, optimizations that reduce the workload are of importance. As our focus are screen recordings from mobile devices, we can exploit some characteristics of the mobile platform. In contrast to most desktop systems, mobile devices have a way more structured and fixed GUI. On desktop platforms, it is common to have multiple resizable windows and applications opened at various positions on the screen. Mobile platforms usually only support to display one application at a time which lowers the effort to check where and which application or activity is present. Besides that, most applications follow the design guidelines of iOS and Android, which gives GUIs a fixed structure and makes them more predictable.

Video Preprocessing

To speed up the detection process we slightly preprocess the video material, as there are many frames without a visible difference. At a frame rate of 20 FPS, we skip 54 % of all frames from our test videos. We compare each frame to its predecessor by subtracting the frames and compare the result against a threshold: similar frames result in very small or zero values.

Definition of Log Events

Events of interest are defined on the basis of GUI states and contain a message that is written to the log entry. For our test runs, we defined a list of 30 events ranging from general user interface (UI) events like ñkeyboard openedò to more specific events like ñWhatsapp chat with Person Aò. Figure 2 shows an example of an event definition of a Whatsapp chat. Every event can consist of different elements: comparison of fixed image areas, search for image parts in the whole frame and comparison of text elements or reading text from the frame.

An event definition can contain multiple fixed areas. These image parts are compared to the same area of every video frame. This exploits the fixed structure of the mobile platform, as it is not necessary to search in various positions on the screen in many cases. The similarity is determined by a perceptual hash function [Bu18]. Perceptual hashes are

rather robust and work with different image scaling or artifacts caused by video compression.

To search for an image part that can appear at any or not a fixed position of the screen we use OpenCVôs [Op18] template matching algorithm. As this step can cost more time in contrast to comparing fixed areas it is more efficient to use fixed areas if possible to define an event.

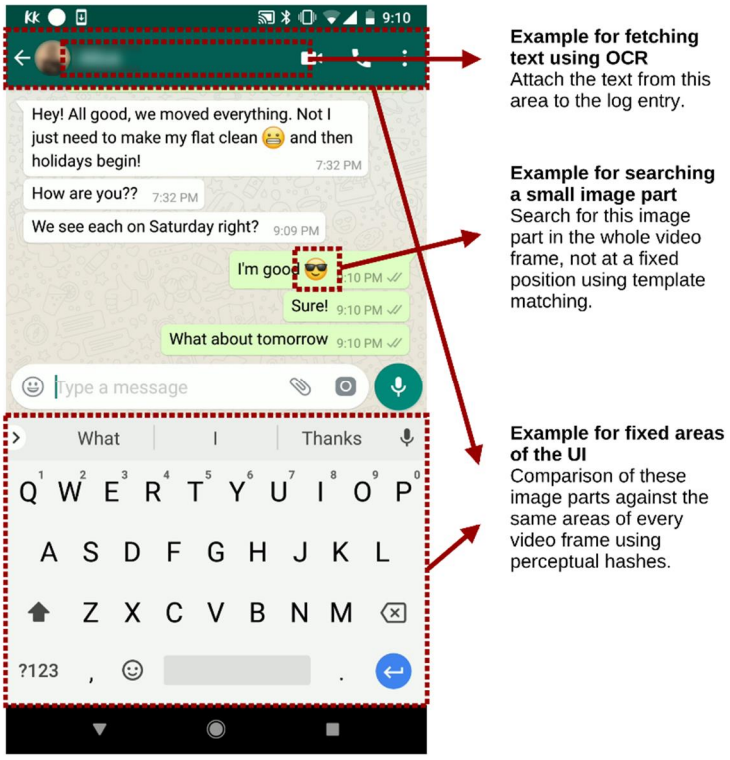

Fig. 2: Definition of an Event. In this example a text string is fetched from an area at the top, two fixed areas are set, one at the bottom and one at the top (keyboard and chat header) and we search for a non-fixed image part in the whole frame (smiley)

We are searching for text strings or reading text from a certain area of the frame using optical character recognition (OCR). For this, we use the open source library tesseract [Te18]. Text recognition is helpful to distinguish a variety of events. An example for fetching text from a frame is the name of a chat partner that is attached to the log message (Fig. 2).

Detecting Events

The detection process works frame by frame and in a certain order, that keeps the computational effort low. Every frame is checked against the list of events. A frame can contain multiple events or no events. The first step is to compare all fixed areas from the event list against the frame, as this step is fast, compared to template matching and OCR. After that, template matching is executed followed by OCR as the last step. As each frame is processed individually, it is possible to scale up the detection process on multiple CPU cores.

3.3 Log files

The resulting log files contain one or more entries for every video frame. An entry consist of frame number, timestamp and log message. To further work with these entries we transfer these logs into a different formatting, to make further analysis more feasible. The final log files are event based instead of frame based. E.g. if 85 consecutive frames contain the event ñInstagram feed openedò, this would lead to one event ñInstagram feed openedò with a timestamp and a certain length. Listing 1 shows example log entries from the resulting final log file. The last log entry for example indicates that a Whatsapp chat to a certain person was opened for 31,4 seconds at 17:37:14 on the 14th of December 2017.

```
1513269377027.0;2017-12-14
17:36:17.027000;2000.0;whatsapp, chatlist active, number
of chats with new messages - 1

1513269377227.0;2017-12-14
17:36:17.227000;1800.0;whatsapp, chatlist opened

1513269379027.0;2017-12-14 17:36:19.027000;800.0;whatsapp
chatlist, scrolled down

1513269380027.0;2017-12-14
17:36:20.027000;3400.0;whatsapp, chatlist active, number
of chats with new messages - 1

1513269380027.0;2017-12-14
17:36:20.027000;3400.0;whatsapp, chatlist opened

1513269383627.0;2017-12-14
17:36:23.627000;12200.0;whatsapp chat opened - Luca

1513269386027.0;2017-12-14
17:36:26.027000;6200.0;keyboard opened

1513269394027.0;2017-12-14
17:36:34.027000;2000.0;keyboard opened
```

```
1513269395627.0;2017-12-14
17:36:35.627000;200.0;homebutton pressed

1513269396027.0;2017-12-14 17:36:36.027000;2400.0;home
screen present

1513269398427.0;2017-12-14
17:36:38.427000;32200.0;Instagram opened

1513269398427.0;2017-12-14
17:36:38.427000;32200.0;Instagram, feed opened

1513269398427.0;2017-12-14
17:36:38.427000;32200.0;Instagram, feed opened, unread
messages

1513269434427.0;2017-12-14
17:37:14.427000;31400.0;whatsapp chat opened – Luca
```

List. 1: Example entries from the result log file. All entries contain a timestamp, a length in milliseconds and a log message that describes the event

4 Results

Altogether 79790 frames were processed from the 68 minutes of video material and a log file with one or more entries per frame was generated. This video frame based log file was transferred into a different formatting as described above (see listing 1). In total, the final log file consisted of 931 correctly detected events. The result log file did not miss any events but contained 1.3 percent false positive events using the originally collected high-quality video material. We define a false positive as an event with a correct log message that did not occur in the video material. The most problematic event definition was the event ñcalculator openedò. The calculator was only opened one time in all videos but was detected falsely 11 times. The reason for these false results was an imprecise definition of the calculator event. Another reason for several messed up log messages were events that fetched text from transition frames using OCR. On these frames between switching or closing applications it is not clear which application is present, as both are morphed into each other. These animation frames caused messed up names of chat partners when the chat applications were closed, for example.

Figure 3 illustrates a usage session containing several application usage sessions. We visualize the result log files in reference to van Berkel et al. [Be16] as usage sessions containing detailed application usage sessions. As an application session, we take the time from opening and using an application until the application is closed again or switched. The upper timeline shows application events, the lower blue line shows the use of general

UI events, which can occur in parallel to in-app events. In this example, the events ñkeyboard openedò and ñk key pressedò occurred several times.

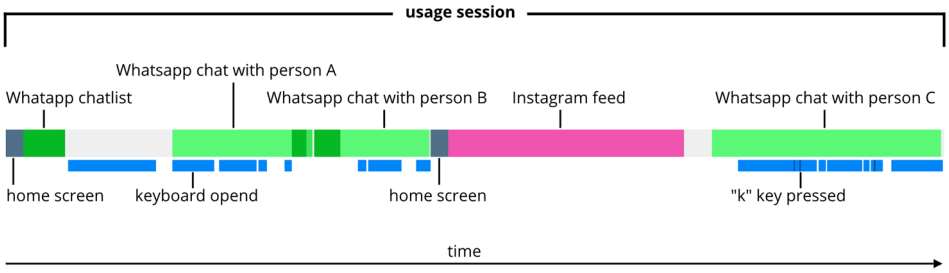

Fig. 3: Visualization of a phone usage session containing the use of different applications and in-app interactions

This usage session shows the use of two different applications (Whatsapp and Instagram) and contains several detailed in-app events about the use of Whatsapp. Three chats with different partners were opened, partly together with the presence of the keyboard. At least in the last chat, the keyboard was opened and used. Most likely in the other cases as well, but for testing we only defined an event for pressing the k key.

Most test events were connected to chat communication. In addition several events regarded the general UI, news applications, and Android system events such as lock screen interactions or displaying the "all applications" menu for example. The result log file contained detailed information about chat communication in four different applications with a chat function: Whatsapp, SMS chats, Instagram and Pinterest. For all chat application events the name of the chat partners was saved. In total there were chats with 13 different persons, the most conversation happened through Whatsapp. One chat partner occurred in all four applications. The event definitions for Whatsapp were most detailed, ranging from taking a picture in a chat (happened two times) to scrolling down the list of chats and number of new messages. Besides those two phone calls were registered to the same person, one missed call and one outgoing call. Other events included using the phoneôs contact list, creating a new contact, viewing the ñtop storiesò in the BBC news application, using the calculator, and whether the device is used in unlocked or locked screen mode.

All event definitions besides one used at least one fixed area and perceptual hash comparisons. Template matching was defined in two events and OCR in 12 events. In general comparing fixed areas worked reliable for our purpose, as well as template matching. The OCR results were very accurate, unless used on transition frames during application switching or closing. We used multiprocessing on a dual-core Intel i7 for all test runs. Processing all 118 high-quality videos took about 10 hours. Ideas for speeding up the process are briefly outlined in the discussion.

5 Discussion

Most data sources for Learning Analytics rely on log files from LMSs or related systems. These data sources have two clear limitations: log events cannot be individually defined in most cases, unless the source code is modified and most the log files are limited to interactions within the system. We address this challenge by introducing a logging approach that relies on the visual screen output. Theoretically, it is possible to track everything that is visible on the student's device screen, leading to data about interactions in very high detail. Log events can be detected in multiple applications while only using one method to generate the log file which eliminates the need to merge log files from multiple sources. Instead of implementing log commands at development stage, researchers can define events of interest based on GUI states instead. This can be done before collecting video data or even after. In case that the logged events are not useful or predictive for the research question, event definitions can be adjusted and the video material can be analyzed again from a different perspective. Another possibility resulting from the use of this approach is that no access to the source code of the applications is required to implement log commands. This opens a new way to log in third party applications, since many applications are closed and do not produce any log files.

5.1 Privacy and Ethical Challenges

A serious challenge before applying our approach in a real-world scenario is possible, is the privacy of studentôs personal data. The studentôs device screen is permanently recorded, saved, and analyzed. This requires an informed consent by participants and raising the question of how participants in research studies can be informed what data they share for what purpose and how they can control what they share. We follow the Privacy by Design [Ca11] approach in order to develop an effective concept for the protection of the privacy of participants already during development. A step in this direction would be that the screen recordings are not be accessible to researchers and would have to be approved by users to be searched for events. Additionally, the anonymization of the resulting log files is obligatory, before they could be used in a Learning Analytics context. With regard to these serious obstacles to privacy, ethical questions also arise as to whether the gathering and use of these log files justifies such a data collection and analysis. This ethical perspective will depend greatly on the success and implementation of a sustainable privacy concept, that ensures the privacy rights of students or participants and is in accordance to current regulations.

5.2 Technical Challenges

For further analysis of learning paths, student behavior or predictions based on these log files, it is essential, that the data quality and correctness is ensured. The result logs of our sample video material reached a high level of correctness for 30 test events, ranging from

very short events like pressing a key on the keyboard to reading in a news app. We identified some event definitions that produce unexpected or messed up results and incorrect log messages that lead to false positives, though no events that occurred in the test video material were missed.

Another technical challenge besides correct log results is the great file size of the video material and computational effort to process the screen recordings. The 118 sample screen recordings we used for this paper to validate the results of our approach as a data source added up to 2,2 GB. Since the recognition works frame by frame, the recognition process can be considerably accelerated by multiprocessing. Besides scaling the number of CPU cores, it seems reasonable to reduce the video frame rate already at recording time, since 54 % of all frames were duplicates or almost duplicates.

6 Conclusions and Future Work

In this paper, we presented an approach for a new data source in Learning Analytics that works beyond an LMS, in multiple applications and independent of the applications source code. We showed that we could generate detailed log files based on mobile screen recordings and track a range of interactions on mobile devices. We demonstrated examples of how this log data can be further processed to serve as a basis for Learning Analytics scenarios from a new perspective.

The next steps involve the development of privacy measures, performance and result optimizations. It is also necessary to investigate how this data source can be used sensibly, what insights are possible, and how this approach can be reasonable from a viewpoint of research ethics.

References

[Ag14] Agudo-Peregrina, Á.F., Iglesias-Pradas, S., Conde-González, M.Á., Hernández-García, Á.: Can we predict success from log data in VLEs? Classification of interactions for learning analytics and their relation with performance in VLE-supported F2F and online learning. Computers in Human Behavior. 31, 542ï 550 (2014).

[Ah14] Aho, P., Kanstren, T., Räty, T., and Röning, J. Advances in Computers. Academic Press, Aug. (2014).

[Be16] van Berkel, N., Luo, C., Anagnostopoulos, T., Ferreira, D., Goncalves, J., Hosio, S., Kostakos, V.: A Systematic Assessment of Smartphone Usage Gaps. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. pp. 4711ï 4721. ACM, New York, NY, USA (2016).

[Bö11] Böhmer, M., Hecht, B., Schöning, J., Krüger, A., Bauer, G.: Falling Asleep with Angry Birds, Facebook and Kindle: A Large Scale Study on Mobile Application Usage. In:

Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services. pp. 47ï 56. ACM, New York, NY, USA (2011).

[Bu18] Buchner, J. imagehash: A Python Perceptual Image Hashing Module, https://github.com/JohannesBuchner/imagehash, accessed: 18/01/2018.

[Ca11] Cavoukian, A. Privacy by Design - The 7 Foundational Principles, (2011).

[CYM10] Chang, T.-H., Yeh, T., Miller, R.C.: GUI Testing Using Computer Vision. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 1535ï 1544. ACM, New York, NY, USA (2010).

[Fr16] Frisson, C., Malacria, S., Bailly, G., Dutoit, T.: InspectorWidget: A System to Analyze Users Behaviors in Their Applications. In: Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems. pp. 1548ï 1554. ACM, New York, NY, USA (2016).

[HBF18] Hepp, Andreas; Breiter, Andreas; Friemel, Thomas N.: Digital Traces in Context. International Journal of Communication, (2018).

[Ki15] Kitto, K., Cross, S., Waters, Z., Lupton, M.: Learning Analytics Beyond the LMS: The Connected Learning Analytics Toolkit. In: Proceedings of the Fifth International Conference on Learning Analytics And Knowledge. pp. 11ï 15. ACM, New York, NY, USA (2015).

[MGF13] Matejka, J., Grossman, T., Fitzmaurice, G.: Patina: Dynamic Heatmaps for Visualizing Application Usage. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 3227ï 3236. ACM, New York, NY, USA (2013).

[Op18] OpenCV library. https://opencv.org/, accessed: 24/01/2018

[PE14] Papamitsiou, Z., & Economides, A. (2014). Learning Analytics and Educational Data Mining in Practice: A Systematic Literature Review of Empirical Evidence. Educational Technology & Society, 17 (4), 49ï 64.

[PK11] Pardo, A., & Kloos, C. D.: Stepping out of the Box: Towards Analytics Outside the Learning Management System. In Proceedings of the 1st International Conference on Learning Analytics and Knowledge. pp. 163ï 167. ACM, New York, NY, USA (2011).

[Te18] tesseract-ocr. https://github.com/tesseract-ocr, accessed: 24/01/2018

[TRG15] Tempelaar, D.T., Rienties, B., Giesbers, B.: In search for the most informative data for feedback generation: Learning analytics in a data-rich context. Computers in Human Behavior. 47, 157ï 167 (2015).