UBC Arts ISIT
Work Learn S19: Learning Analytics/Visual Analytics
Supervisor: Sanam Shirazi
Student: Yan Shao (Shenia) Tung
August 12, 2019

Progress Dashboard for edX MOOC Students - A Proof of Concept

Overview

Currently, the University of British Columbia (UBC) lists over thirty active and archived Massive Open Online Courses (MOOCs) on the edX online learning platform. MOOCs have, on average, a high dropout rate of 95% [1]. As suggested by Davis et al. in their study [2], the unique non-degree, flexible and inexpensive nature of MOOCs likely contributes to a lack of motivation to complete the course. The purpose of this project is to tackle this challenge by developing a student-facing application that informs students of their engagement with course contents, enhances self-motivation in course completion, and improves course completion rates in UBC MOOCs offered on edX.

Solution

This project attempts to replicate a past solution designed by Davis et al. [2], with a modified set of metrics selected from correlated variables as shown in studies done by Pursel et al. [3] and Macfeyden et al. [4].

Dashboard Design

Davis et al. hypothesized that compared to dashboards with an isolated view of their own behaviors, learners will benefit more from dashboards that shows “comparison of their own behavior with that of successful peer learners” [2]. They developed the Learning Tracker widget in the same study [2], which consists of a radar chart with six student engagement metrics and two plots, one plotting the course engagement of a current student, one plotting the course engagement of a previously successful student.

UBC Arts ISIT
Work Learn S19: Learning Analytics/Visual Analytics
Supervisor: Sanam Shirazi
Student: Yan Shao (Shenia) Tung
August 12, 2019

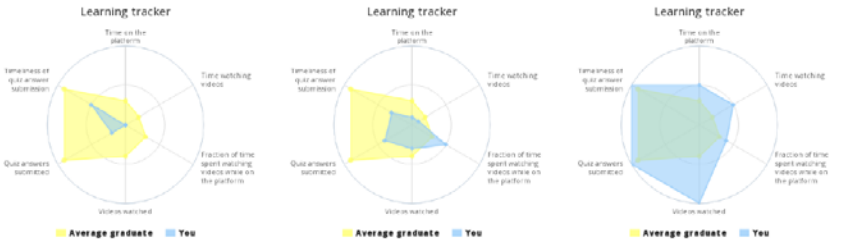

Figure 4: Three versions of our Learning Tracker widget with data from week 9, one showing a learner who dropped out early in the course (left), one who dropped out in the middle of the course (middle), and one graduate who is highly engaged with the course (right).

Figure 1: Davis et al.'s demonstration in their 2010 study [2], "Encouraging Metacognition & Self-Regulation in MOOCs"

In their study, Davis et al. deployed the demonstrations to students in a MOOC on the edX platform [2]. Students were divided into two groups, one had access to the Learning Tracker widget while the other one did not. After the course, the experiment result showed that there is a significant difference between the two groups on two particular metrics – number of quizzes submitted, and timeliness of quiz submission. And that there is an overall increase in the percentage certificate-earning learners compared to a previous run of the same course. This project finds this finding valuable and thus adapts Davis et al.'s widget design [2], with a modification on the metrics used.

Metric Definition

According to the findings of Pursel et al. [3], significant predictors of MOOC completion include students':

- level of education,
- number of video views,
- number of forum posts and comments, and
- pre-course survey on intended behaviors

Of these, video engagement and forum engagement are considered student engagements in the course relevant to this project. Since quiz completion is a significant contributor to course completion, coupled with evidences from the study by Davis et al, [2] it is of interest to document quiz completion progress as well. This project thus document student engagement in three categories - Video, Discussion, and Assessment, onto the radar chart-based widget design. For the purpose of clean layout, two to three metrics are picked for each category.

The project first attempts to identify metrics for the Assessment category. With the limited number of graded assessments, this project measures students' engagement based on the quizzes submitted. In the study by Davis et al., [2] results showed a significant difference between participants placed in different groups on two metrics:

- the fraction of quizzes completed (number of quizzes completed/number of quizzes available)
- the average of quiz submission timeliness (the average number of hours between the completion of each quiz and each quiz deadline)

where students who have access to the dashboard showed a higher score in both metrics (and no significant difference in the other four metrics). It is of interest to investigate if these two are significant indicators of student engagement in an online learning platform, and if displaying these metrics on a student dashboard will have a positive effect on student learning. Therefore these metrics are chosen for the Assessments category of metrics.

The Discussion category quantifies student's engagement with peers and instructors on the discussion forum. In a study by Wise et al., [5] it is suggested that the depth with which students attend to posts, the breadth with which they attend to others' comments, temporal contiguity of their listening, and reflective activity in a previously viewed post are good indicators to strong student engagement. In the context of this project, student engagement in discussions will be evaluated based on the following:

- depth of discussion posts (the average number of comments students posted under each thread)
- breadth of discussion posts (the number of threads students posted under)
- number of posts viewed

There are limited number of studies done to investigate a set of metrics for students' engagement with videos, however, many studies (Wu et al, 2018 and Bonafini et al, 2017) [6, 7] and marketing guides bases user video engagement on the number of videos watched. We wish to also investigate how engaged students are with the videos by the number of re-watches, which may indicate how closely, or in depth, students pay attention to the videos. Thus we use the following metrics:

- the fraction of unique videos watched (the number of videos watched/the number of videos available)
- the fraction of unique videos rewatched (the number of videos rewatched/the number of videos available)

Scope

Due to limited scope of the project, the project limits its users to students of the current iteration of an Asian Studies course offered by UBC on edX. The project limits its past student group to one previous iteration of the same course, also offered by UBC on edX. The project runs locally on address 127.0.0.1:1000

For future replications and expansion, developers may wish to modify the code to include edX courses with past iteration(s) of similar topics, contents and layouts. Developers may also wish to modify the code to deploy this project.

Program Interface

The product of this project is a web dashboard that displays student data on the following metrics,

- the fraction of quizzes completed, and
- the timeliness of the quiz submissions
- Depth (the average number of comments students posted under each thread)

- Breadth (the number of threads students posted/posted under)
- Number of posts viewed
- the fraction of unique videos watched
- the fraction of unique videos rewatched

The dashboard contains 5 user input fields: username, module list, normalization option, graph refresh button, and bar chart toggle.

Initially, the program requires the user to fill in the student's username on edX, a list of course modules (chapters) the student wants information on, and a normalization option (Figure 3). When the user fills in all three fields and clicks the graph refresh button, a radar chart is filled out (Figure 4). [Re-number figures]

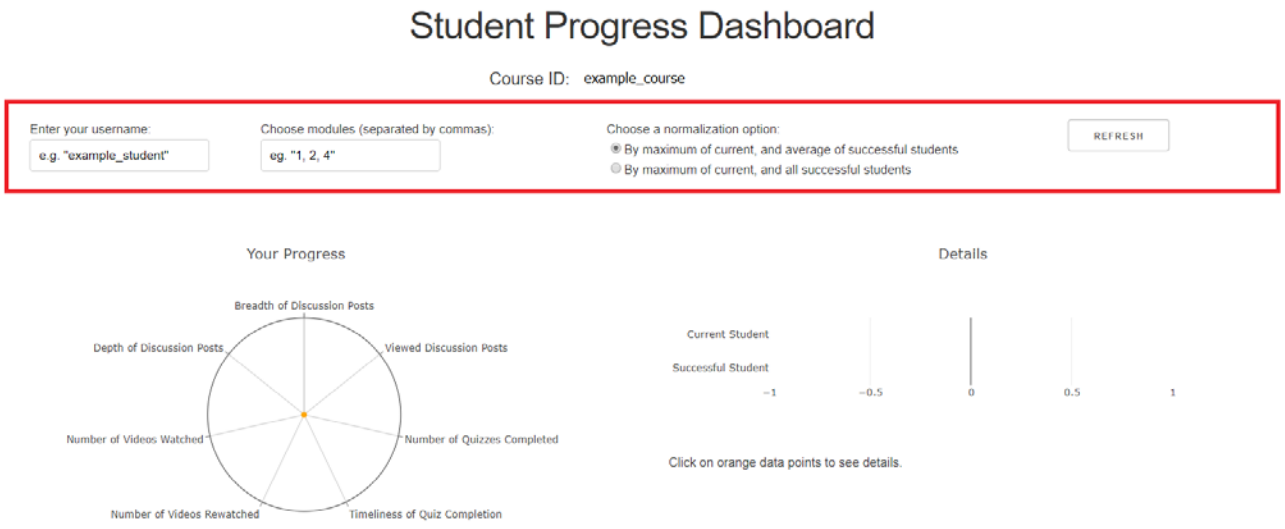

Figure 2: Initial Interface - username, module list, normalization option, graph refresh

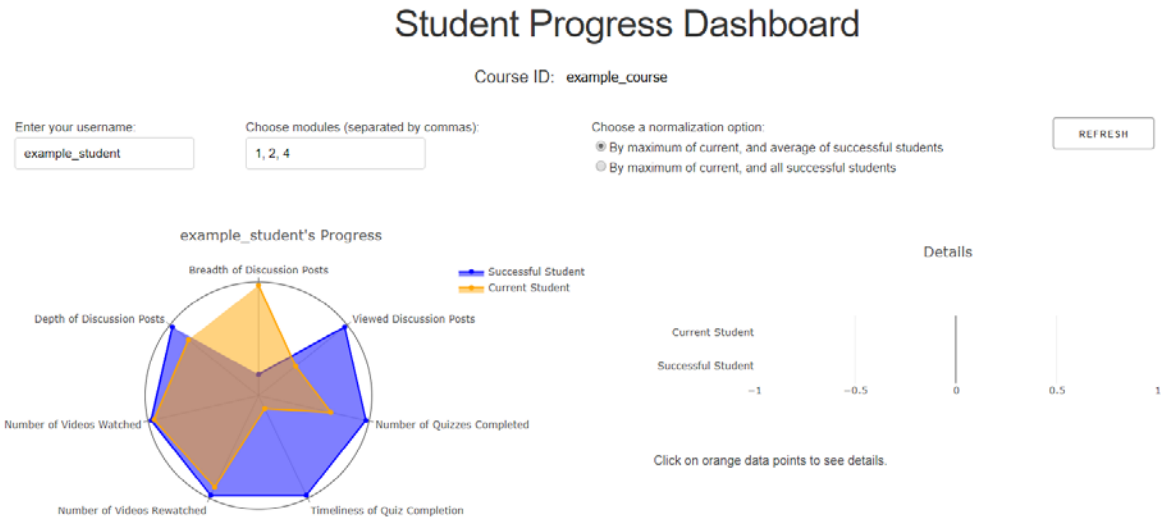

Figure 3: Refreshed Radar Chart

The orange plot shows the user's data, while the blue plot shows the progress of a past student who has taken and successfully completed the same course in a previous iteration.

The Bar Chart Toggle is visible when the user hover their pointer on any one of the orange data points (Figure 5). When clicked, a bar chart shows more detail on the selected metric (Figure 6).

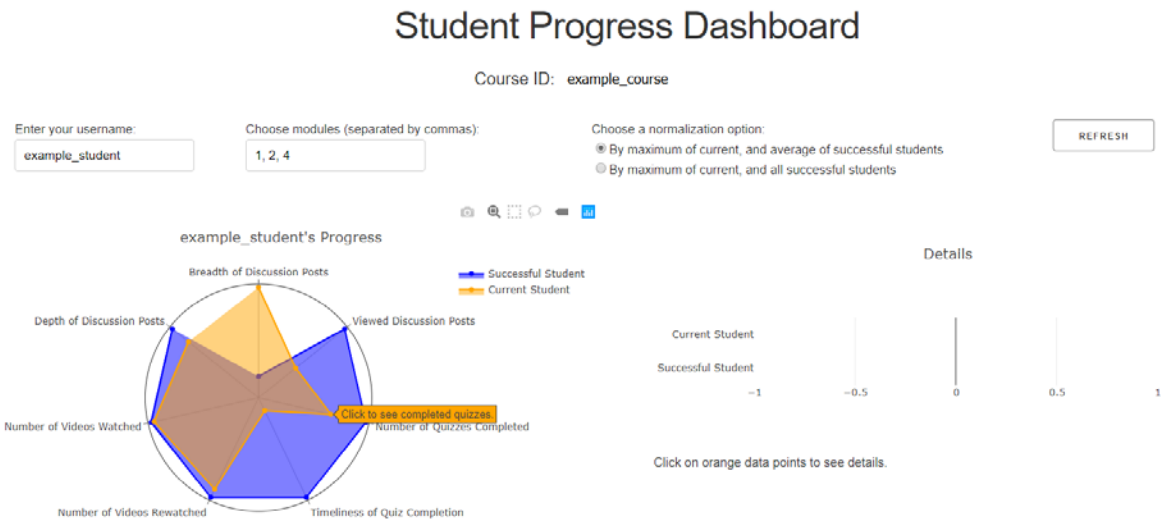

Figure 4: Bar Chart Toggle

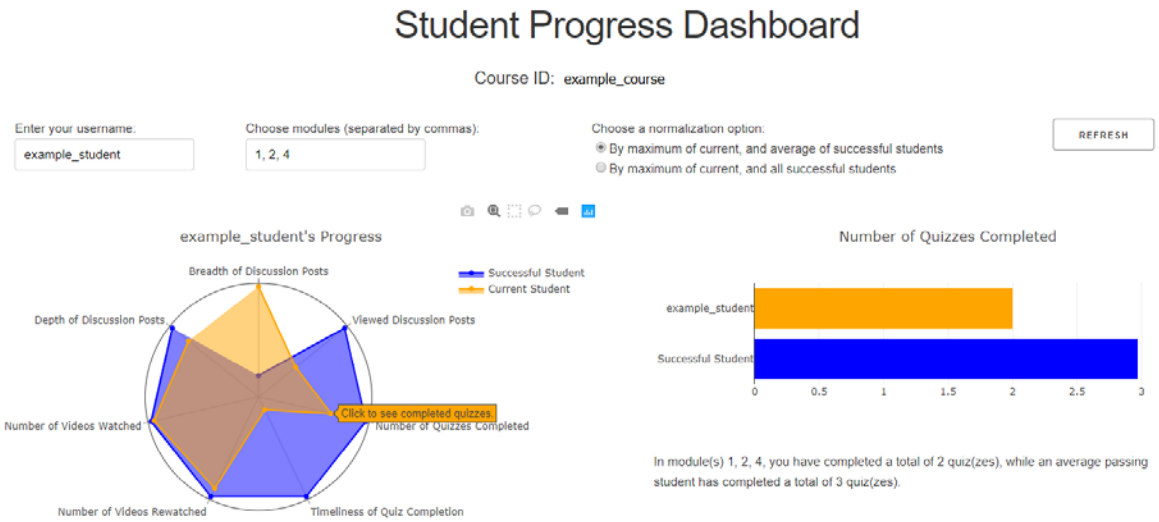

Figure 5: Bar Chart

To see progress on different modules, simply re-enter the list of modules and click on graph refresh (Figure 7).

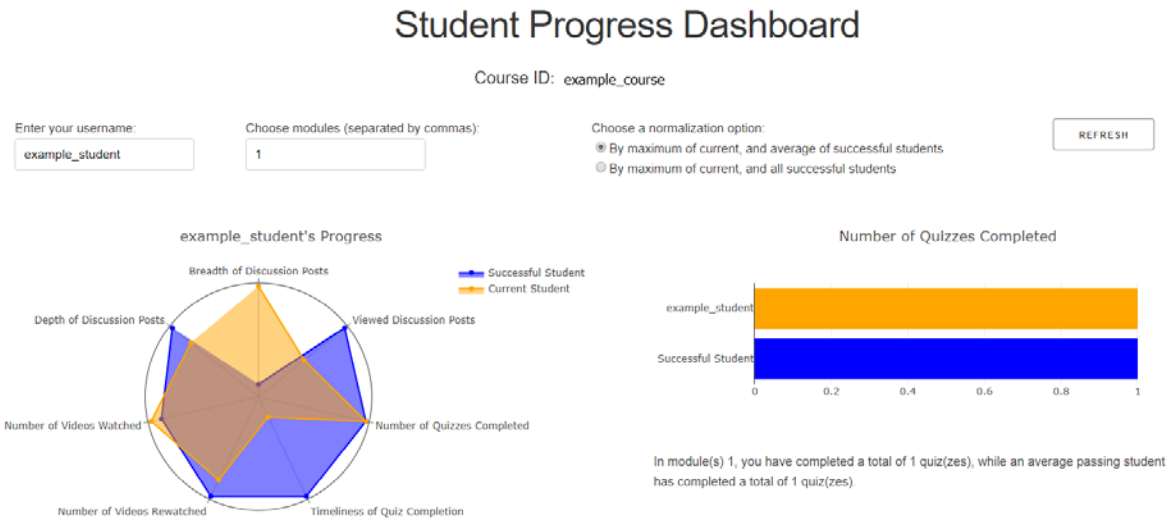

Figure 6: Select Modules

Users may also select different normalization methods. When the graph is displayed, the user's data in each metric is divided by the maximum value in each metric of the user and other successful students (the normalizing factor). The users may choose different normalizing factors, between the maximum values out of the user and the average of successful students, or the maximum value out of the user and all successful students (Figure 8).

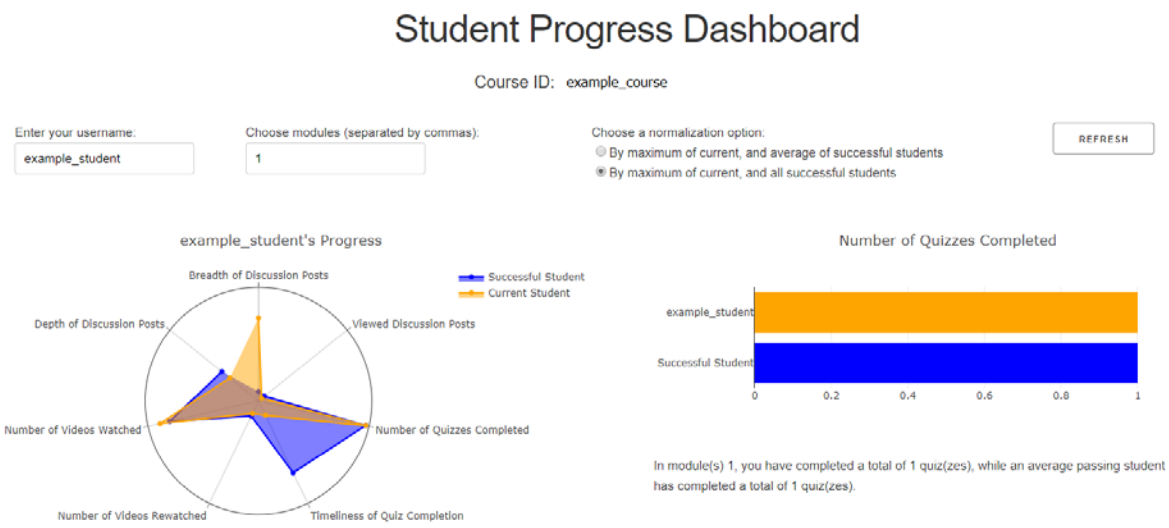

Figure 7: Select Normalization Option

Both radar chart and bar chart are refreshed when the graph refresh button is pressed. The radar chart is refreshed from the most recent username, module list and normalization option inputs, and the bar chart is either left blank if there is no previous bar chart toggle input, or refreshed from the most recent username, module list, and bar chart toggle input.

Input and Output Files

The program requires access to the edX Data Package, a collection of data including course information and student activities in courses. UBC has stored the edX Data Packages for its edX courses on Google Bigquery.

For this project, relevant data tables on Google BigQuery is first read into Tableau through custom, aggregated queries, then downloaded as csv files with Tableau's data export option. For all current users, four files in total are exported - one containing all users' data video and quiz metrics, one containing all users' data on discussion post authorship, one containing all users' data on discussion post views, and one containing all discussion post data including the unique post id and title. For all past students, four files of the same format as above are exported. For both past students and current users, the discussion post data files are manually appended with module columns, specifying the module that that particular post is related to.

This program takes the above files in MS Excel Workbook format as inputs, as well as user inputs specified in the Program Interface section. This program cleans, reformats and aggregates input Workbook files by unique user id and module pair, and outputs CSV files containing user information and metric data.

The exact format of each input and output file, including title, fields and file format, can be found in Appendix B.

UBC Arts ISIT
Work Learn S19: Learning Analytics/Visual Analytics
Supervisor: Sanam Shirazi
Student: Yan Shao (Shenia) Tung
August 12, 2019

Program Structure

The program consists of the following files:

- README.txt
- \appShared.py
- \CSVs\Video Quizzes Info\current_QV.xlsx
- \CSVs\Video Quizzes Info\past_QV.xlsx
- \CSVs\Discussion Posts\current_D.xlsx
- \CSVs\Discussion Posts\past_D.xlsx
- \CSVs\Discussion Views\current_DV.xlsx
- \CSVs\Discussion Views\past_DV.xlsx
- \CSVs\Discussion Mapping\current_DM.xlsx
- \CSVs\Discussion Mapping\past_DM.xlsx

The program requires the following data services and software:

- edX Data Package
- Google BigQuery
- Tableau Desktop
- edX Insights Courseware
- Python

SQL queries are required for some parts of the program. All SQL queries are stored in the following files:

- \CSVs\Video Quizzes Info\student_video_quiz_current.sql
- \CSVs\Video Quizzes Info\student_video_quiz_past.sql
- \CSVs\Discussion Posts\posts.sql
- \CSVs\Discussion Views\postViews.sql
- \CSVs\Discussion Mapping\discussionMapping.sql

This program produces the following output files:

- \CSVs\Python Outfiles (raw)\rawcurrent.csv
- \CSVs\Python Outfiles (raw)\rawpast.csv

For specific format within each excel file, see Appendix B.

Initial Setup

First, select relevant MOOC(s) and some of its past iterations. It is required that the course has at least one past iteration with similar topics and content for each module. The scope of this project is limited to the particular

Asian Studies course, the user group limited to current students, and past reference group limited to past students of a previous iteration of the same course.

Create a Google BigQuery account and request edX Data Package for relevant MOOCs with edX staff. Request edX Data Package to be stored to Google BigQuery account.

In Tableau Desktop, create a workbook. Set up Data Source to connect to Google BigQuery account to access edX Data Package.

Set up Custom SQL queries using, one at a time, each SQL file above and run query. Using Tableau's Export Data to CSV option, export queried data into CSV files of the same names as each SQL file. Convert each CSV file into XLS or XLSX format.

Access discussion forums of the relevant courses through the edX Insights Courseware []. For the past_DM.xlsx and current_DM.xlsx files, create a 'module' column and manually map each discussion post row to a related module using discussion forum tags.

For the most up to date data, set up refresh schedules when requesting edX Data Package data, and run SQL queries, discussion post/module mapping regularly.

SQL Files

- \CSVs\Video Quizzes Info\student_video_quiz_current.sql
- \CSVs\Video Quizzes Info\student_video_quiz_past.sql
- \CSVs\Discussion Posts\posts.sql
- \CSVs\Discussion Views\postViews.sql
- \CSVs\Discussion Mapping\discussionMapping.sql

The SQL files student_video_quiz_current.sql and student_video_quiz_past.sql select

- course_id, user_id, username, module
- numQuizzesCompleted (number of quizzes completed by the student in a particular module)
- numQuizzesAvailable (number of quizzes available in a particular module)
- quizTimeliness (amount of time, in hours, between the student completing the quiz and the quiz deadline)
- numVidUnique (number of videos in a particular module)
- numVidUniqueWatched (number of videos watched by a student in a particular module)
- numVidRewatches (number of videos rewatched by a student in a particular module)

from the tables

- person_course
- course_item
- person_problem

- video_axis
- video_stats_day

The files select these information form the tables stored in each course folder in the Google BigQuery account. Note, for the file student_video_quiz_past.sql, the query selects only information on the students who have successfully completed the course.

The SQL file posts.sql selects,

- course id (identifies which iteration of the course the post belongs to),
- username (post author),
- slug_type (type of post: initial_post, response_post or comment),
- slug_id (unique post id),
- thread_id (thread id of the thread the post belongs to, same as slug_id if it is an initial_post), and
- title (title of the thread the post belongs to).

The files select these information from the table forum_posts stored in each course folder in the Google BigQuery account.

The SQL file postViews.sql selects

- course_id,
- username, and
- thread_id

where forum_action is read, from the table forum_events stored in each course folder.

The SQL file discussionMapping.sql selects

- course_id,
- slug_id, and
- title

from the table forum_posts stored in each course folder.

Program Execution

Run appShared.py. This script imports, cleans, reformats and aggregates input Workbook files by unique user id and module pair, and outputs CSV files containing user information and metric data. This script also produces the interactive dashboard above. The interactive dashboard runs locally on address 127.0.0.1:1000

This scripts uses an external stylesheet for the interactive dashboard. The script uses python libraries dash, dash_core_components, dash_html_components, plotly.graph_objects, pandas, numpy and math.

UBC Arts ISIT
Work Learn S19: Learning Analytics/Visual Analytics
Supervisor: Sanam Shirazi
Student: Yan Shao (Shenia) Tung
August 12, 2019

Initial Load

Data Import

appShared.py reads the following files:

- \CSVs\Video Quizzes Info\current_QV.xlsx
- \CSVs\Video Quizzes Info\past_QV.xlsx
- \CSVs\Discussion Posts\current_D.xlsx
- \CSVs\Discussion Posts\past_D.xlsx
- \CSVs\Discussion Views\current_DV.xlsx
- \CSVs\Discussion Views\past_DV.xlsx
- \CSVs\Discussion Mapping\current_DM.xlsx
- \CSVs\Discussion Mapping\past_DM.xlsx

Data Cleaning, Reformatting and Joining

appShared.py cleans imported data by dropping rows with null username or module cells, or rows with module cells not labeled numerically (e.g. as 'Course'), from user-centric datasets. For datasets with quiz completion columns, appShared.py fills null cells with value zero. For datasets with video watch columns, appShared.py fills null cells with the mode value of the column. For datasets with discussion columns, appShared.py fills in null thread_id cells with slug_id cells for rows with slug_type as initial_post, (as thread_id is the slug_id when the post is the initial post of a thread) and drops all other rows with null thread_id cells and rows with null slug_id cells. For datasets with discussion metric columns, appShared.py fills null cells with value zero.

Since it is unclear how edX classified a video as watched, for rows with null video watch value, the program fills the cells with the mode value of the metric. For rows with null quiz completion value, the program fills with value zero because if quiz completion is null, it is due to edX not having any record of quiz answer submission. Thus the value is null and should be replaced with value zero. For rows with null discussion metric value, the program fills the cells with value zero because if discussion metric is null, it is due to edX not receiving any record of discussion post submission.

The program then reformats cells to have appropriate datatypes, and joins all datasets of the same iterations by left joining, with student_video_quiz on the left. These tables are joined on one of username and user_id, and module. student_video_quiz is on the left in the left join because the program only wants to keep successful student records for past iterations. While student_video_quiz contain data on only students who have completed the course, other datasets contains data on all students, which the program filters by left joining.

Data Export

The program exports one file per iteration, as rawpast.csv and raw[current].csv

Each file has primary key (course_id, user_id, username, module). For each registration unique on (course_id, user_id, username), the file has a row of record for their performance in each module.

Each file also contains the following columns:

- numQuizzesAvailable (see details in the SQL Files section),
- numQuizzesCompleted (see details in the SQL Files section),
- quizTimeliness (see details in the SQL Files section),
- numVidRewatches (see details in the SQL Files section),
- numVidUniqueWatched (see details in the SQL Files section),
- numVidUnique (see details in the SQL Files section),
- numUniqueThreads (number of threads the user writes under in the specified module),
- numUniquePosts (number of posts the user posts in the specified module),
- numPostViewed (number of posts the user viewed in the specified module)

Requesting User Input/Loading Dashboard

After the initial load, users are then required to enter the inputs specified in the Program Interface section.

Data Aggregation

When the graph refresh button is clicked, the callback function button_refreshGraph feeds input information into the function normalize(). The function normalize() selects data of relevant modules for current user in raw[current] and all successful users in raw[past], and calls retSuccessfulStudent() and retCurrStudent().

The function retSuccessfulStudent() calculates the sum of all relevant modules for each successful student in each column listed in the Data Export section, and calculates the following aggregate columns across modules:

- quizPercentCompleted (numQuizzesCompleted/numQuizzesAvailable)
- vidPercentWatched (numVidUniqueWatched/numVidUnique)
- avgTimeliness (quizTimeliness/numQuizzesCompleted)
- vidPercentRewatched (numVidRewatches/numVidUnique)
- avgResponsePerThread (numUniquePosts/numUniqueThreads)

The function retSuccessfulStudent() returns a data frame (metricpastSuccessfulAggregatedDF) of all students with the columns listed in the Data Export section (summed across modules) and the above metrics, and a data frame (metricspastSuccessfulAvgDF) of one row - the average value of all columns (averaged across students) in the data frame.

The function retSuccessfulstudent() calculates the sum of all relevant modules for the current student in each column values in the Data Export section, and calculates the aggregate columns listed above. The function returns a data frame (metricscurrentAggregatedDF) of one row - the current student with the columns listed in the Data Export section (summed across modules) and the above metrics.

The column values of the data frame metricspastSuccessfulAvgDF is used to plot successful student progress, while the column values of the data frame metricscurrentAggregatedDF is used to plot current student progress.

UBC Arts ISIT
Work Learn S19: Learning Analytics/Visual Analytics
Supervisor: Sanam Shirazi
Student: Yan Shao (Shenia) Tung
August 12, 2019

Data Normalization

Currently, the metrics of the radar chart are in different units: percentage (strictly less than 1, for video metrics and quiz completion metric), hours (for quiz timeliness metric), number of posts (for viewed posts metric), number of threads (for breadth of discussion metric), and ratio of post and thread (strictly greater than 1, for depth of discussion metric). Normalization is required for ease of readability (to prevent severe skewedness of the plot) and to establish unbiasedness in showing the recommended progress needed in each metric to ensure successful course completion

One method is to take the maximum values in each column across the metricspastSuccessfulAvgDF and metricscurrentAggregatedDF data frames as a series, then divide the values in metricspastSuccessfulAvgDF and metricscurrentAggregatedDF data frames by the maximum series.

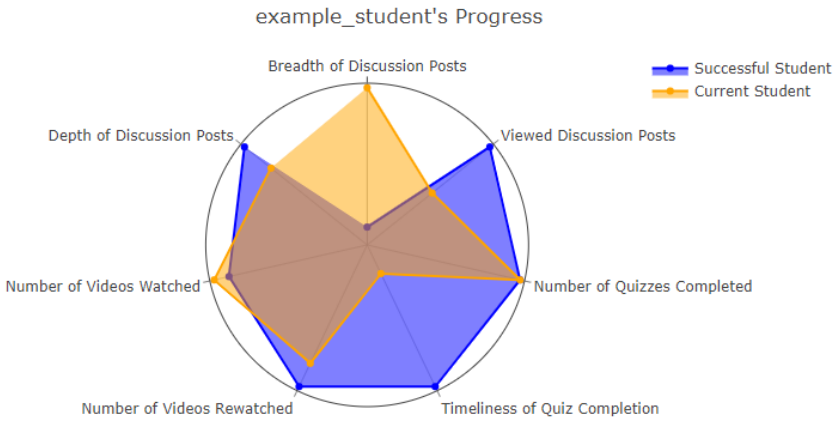

Figure 8: Normalize by maximum between current student value and average of successful student values

Another method is to take the maximum values in each column across the metricspastSuccessfulAggregatedDF (which contains all successful student data) and metricscurrentAggregatedDF data frames as a series, then divide the values in metricspastSuccessfulAvgDF and metricscurrentAggregatedDF data frames by the maximum series.

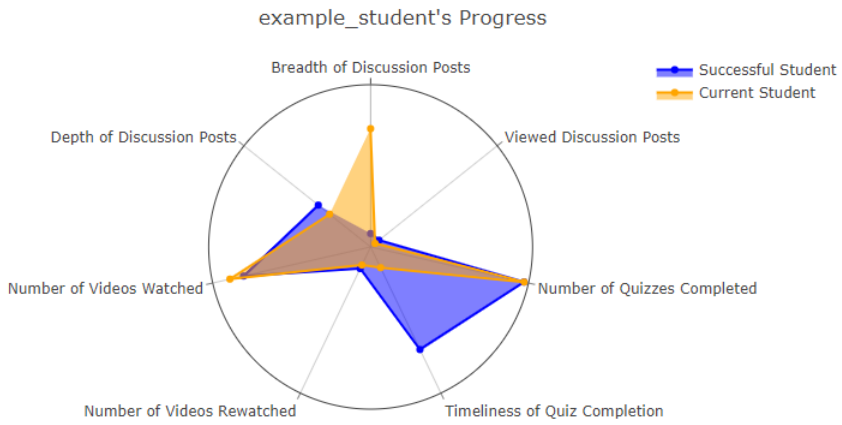

Figure 9: Normalize by maximum of current student value and all successful student values

The normalized values are then displayed on the radar chart.

The differences in the methods is that in the creation of the normalization factor (maximum series). One takes the maximum between 2 values - the average value of past students and the value of the current student, while the other method takes the maximum between multiple values – the values of past students and the value of the current student.

Stylesheet

This script uses an external stylesheet by user Chris P on CodePen to create the dashboard. See References for url.

Dashboard Layout

The interactive components of this dashboard include username input, module list input, normalization option input, graph refresh button, radar chart, and bar chart. The layout is supported by the external stylesheet mentioned in the previous section, and by the dash, dash_core_componeents, dash_html_components, plotly.graph_objects, and dash.dependencies library functions.

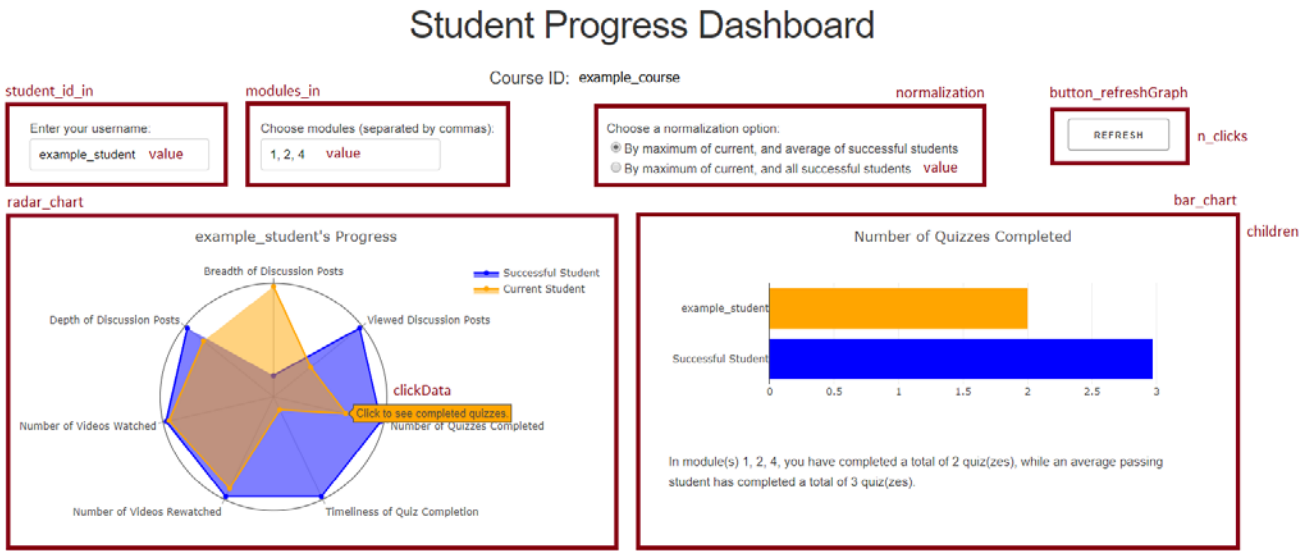

Figure 10: Dashboard Layout, labeled with component ids corresponding to Figure 10

Callback Functions

To support the interactive components of the program, there are two callback functions - update_graph and display_click_data.

When the graph refresh button (button_refreshGraph) is clicked, update_graph updates the radar chart with the value state of normalization (normalization), username (student_id_in), and module list (modules_in). When the graph refresh button is clicked or when the bar chart toggle (radar_chart clickData) is updated, display_click_data updates the bar chart with the state of the username, module list and bar chart toggle input.

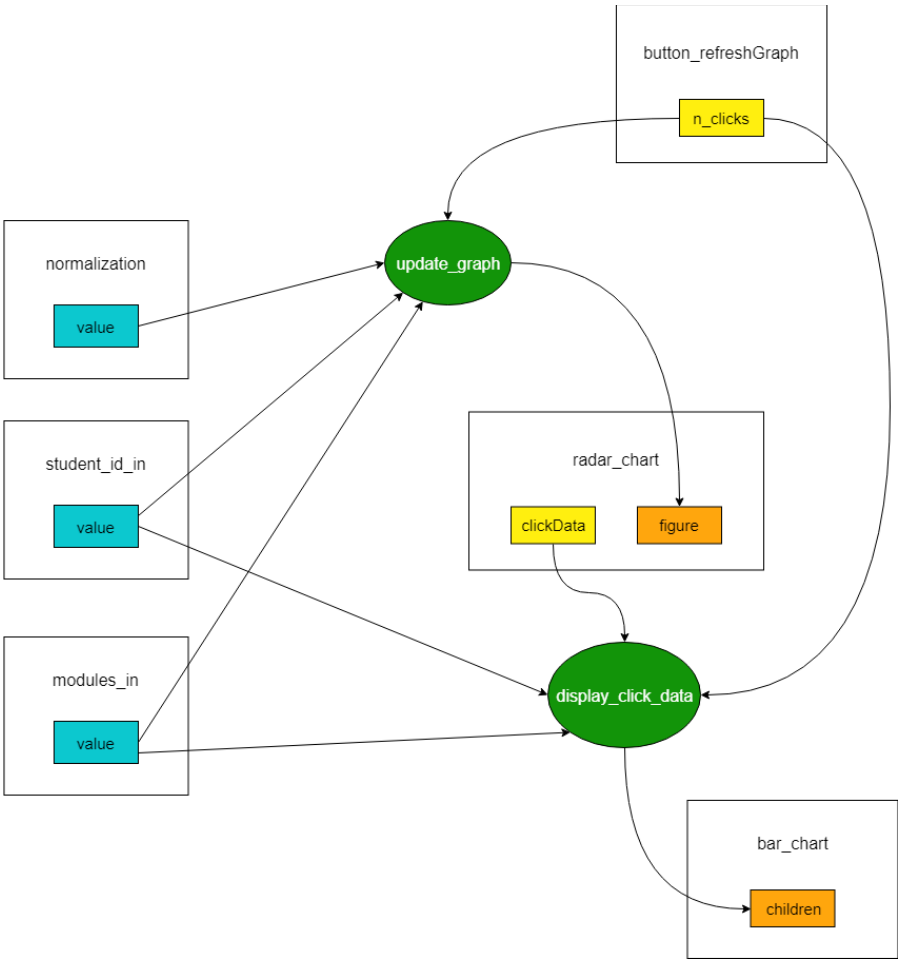

Figure 11: Callback Diagram

UBC Arts ISIT
Work Learn S19: Learning Analytics/Visual Analytics
Supervisor: Sanam Shirazi
Student: Yan Shao (Shenia) Tung
August 12, 2019

Improvements and Extensions

There exist some inefficiencies and areas requiring further research and development in the program that may be considered for revision.

First, due to an authentication failure between appShared.py and Google BigQuery, this program cannot directly load data from Google BigQuery. A potential reason behind this failure is that the Google BigQuery project containing relevant data tables are currently under an unpaid plan, while Google Cloud documentations [] list upgrading to a paid plan as a necessary step for direct loading. This program thus currently requires exporting data from Tableau through a connection between Tableau and Google BigQuery (see Initial Setup). For future revisions, developers may wish to upgrade current account/project with Google BigQuery to a paid version in order to enable direct loading. Alternatively, Google BigQuery allows exporting and downloading SQL query results directly through its classic UI. Future developers may wish to use this function to improve data flow in future revisions.

Second, there is currently no data that maps discussion thread to respective modules in the course. This program currently requires manually mapping discussion titles in csv files to modules by examining discussion thread tags on the discussion forum. For future revisions, developers may wish to scrape data from the discussion forum for this purpose, or find other solutions to automate the process.

Third, there are some discrepancies when interpreting edX Data Package documentation. The program assumes that all registered users are listed as a student in the person_course table, but some users documented in discussion forum activities in the table pc_forum are not listed in the table person_course. A possibility is that some users have participated in the discussion forum but not registered in the course. Another possibility is that the data within edx Data Package tables are incomplete. For the purpose of this project, only students listed in the person_course table are supported for the dashboard. For future development, developers may wish to more closely examine edx Data Package documentations.

Fourth, the dashboard is currently lacking in three areas; lacking detail, lacking authentication, and lacking consideration in normalization methods used. It lacks specific details on how each user is contributing to their progress. For example, the dashboard only shows how many discussion threads the user has participated in (posted under) but not which threads or links to these threads. Ideally, the dashboard should include more information on traceable information on how the user is contributing to their learning progress as displayed on the dashboard. For future revisions, developers may wish to display these details.

The dashboard application lacks authentication in the sense that currently, it allow any user to access all current student progress and an anonymous aggregation of past student progresses. Ideally, the dashboard should have some form of authentication to allow only authorized users (i.e. the user themselves) to access their own progress and an anonymous aggregation of past student progresses. This is a necessary measure to take if this application is to be deployed and should be considered by future revisions.

The dashboard also lacks consideration in normalization methods used for the radar chart plots. Currently, the metrics of the radar chart are in different units: percentage (strictly less than 1, for video metrics and quiz completion metric), hours (for quiz timeliness metric), number of posts (for viewed posts metric), number of threads (for breadth of discussion metric), and ratio of post and thread (strictly greater than 1, for depth of discussion metric). Normalization is required for ease of readability (to prevent severe skewedness of the plot) of the radar chart. For this purpose, the method of dividing the two values in the two plots (current and past) by maximum of the two values in each is used as default. Normalization is also required to establish unbiasedness in showing the recommended progress needed in each metric to ensure successful course completion. Since we do not have research on what the recommended value in each metric is, the method of dividing the two values in the two plots by maximum of the current value and all successful student values in each metric listed as an option. These two methods are used for these considerations, but no reference material has been consulted for this decision. It may be of interest to revise the normalization methods to find one optimal method that satisfy both criteria.

Fifth, the dashboard currently limited to the particular Asian Studies course on the edX platform. This dashboard program is currently not scalable due to time constraints and more essentially, the need to manually map discussion thread to modules. For future revisions, developers may wish to scale this dashboard to include other edX courses (with similar data formatting in the edX Data Package) with at least one past iteration.

Lastly, measures may be taken to improve the readability and structure of code involved in this project to improve efficiencies in queries, data cleaning, reformatting and visualization, as well as readability and reusability of modular code in the program.

Course of Project and Difficulties Encountered

This project began as a loosely-defined pilot project to create a student-facing dashboard that would allow learners to monitor their own progress. To develop some ways to measure and display student progress required first studying literature on student engagement, then defining metrics based on literature and availability of data, before moving onto dashboard design.

There were multiple drafts in the dashboard layout and reviews in between by supervisor and generous peers. One notable hurdle was to find a way to normalize the various metrics so that the radar chart plots are fair representations (i.e. weighs each metric fairly) and do not appear skewed. This continues to be a topic for research.

When building the dashboard, initially, the dashboard is to be built in Python. A Google Service Account is created, and API keys are requested for this project to establish data connection between Google BigQuery and the python notebook script (.ipynb). Due to soon-discovered issues with authentication issue between Google BigQuery and the script (see Improvements and Extensions), the dashboard shifted to the Tableau platform.

UBC Arts ISIT
Work Learn S19: Learning Analytics/Visual Analytics
Supervisor: Sanam Shirazi
Student: Yan Shao (Shenia) Tung
August 12, 2019

Tableau is chosen as the next candidate for data visualization as data connection between Tableau Desktop and Google BigQuery is straightforward and did not require a Google Service Account or paid plans. However, difficulty soon arises as there are no built-in functions in Tableau to support the radar chart component of the dashboard. Traditionally, radar charts in Tableau are built using the Polygon selection from the drop-down menu in Marks panel. This method required pivoting of data from a user-centric format to a metrics-centric format (where each row of data represents one user-metric pair instead of one user with all metrics). Moreover, there is no straightforward method to clean the raw data or overlap two radar chart plots on one set of axes. Data cleaning is essential in working with data, and overlapping radar chart plots are required to support the core concept behind this dashboard, thus (with timely discovery of the data export function in Tableau - see Initial Setup) the dashboard shifted platform again to Python.

To first develop an overall structure of the program, the Python script was first developed in Jupyter Notebook in .ipynb format. The library matplotlib.pyplot was first used to graph the radar chart plots. After an initial success, the program migrated to another text editor and converted into the .py format (using the Download as function in Jupyter Notebook) for ease of dashboard building and debugging.

To develop the interactive dashboard, two libraries are considered – Bokeh and Dash. The library Dash was used for initial ease of matplotlib plot conversion to Dash (function for matplotlib plot conversion to Bokeh has been deprecated, and conversion is no longer supported [8]. The conversion function in Dash was, however, never used, and instead radar chart plots are generated with Dash libraries for ease of adding interactions over the radar chart. Other components including widgets and the side bar chart was developed.

Towards the end of this project, this report is drafted. The time between report submission and the end of this project will be used to develop a web-scraping component in the program to replace the manual labour currently used in matching each discussion thread to a course module. A working-copy of the program with some undocumented progress will be submitted along with this report.

Conclusion and Acknowledgments

In their study done in [past], Davis et al. suggests that from the lack of “pressure from teach/parents, […] financial obligations, [… or …] academic credit on the line,” educators face the challenge of high MOOC dropout rates [2]. MOOC dropout rates of around 95% in the average MOOC [1], has posed challenges to many MOOC-offering bodies. To combat this challenge, this project is established to develop a student-facing dashboard to help students keep track of their MOOC learning progresses in a timely and goal-oriented manner.

This project aims to replicate a past solution designed by Davis et al. [2], with a modified set of metrics selected from correlated variables as shown in studies done by Pursel et al. [3] and Macfeyden et al. [4]. It is completed in four stages; metric definition for dashboard graph components, initial design of dashboard, implementation of dashboard, and final reporting and further improvements. Each stage spanned around one month between May and August 2019. This project is proceeded by Shenia Tung, Learning Analytics Project Assistant, under the

UBC Arts ISIT
Work Learn S19: Learning Analytics/Visual Analytics
Supervisor: Sanam Shirazi
Student: Yan Shao (Shenia) Tung
August 12, 2019

supervision of Ms. Sanam Shirazi, Senior Data Analyst (Learning Analytics) at the Arts Instructional Support and Information Technology department at the University of British Columbia. Guidance and generous advice from Ms. Shirazi and peer reviewers of the dashboard design are greatly appreciated.

This project is a not-yet deployed pilot project, with potentials of improvement for further development. Interests for further development are welcome with discretion of the Arts Instructional Support and Information Technology department at the University of British Columbia.

UBC Arts ISIT
Work Learn S19: Learning Analytics/Visual Analytics
Supervisor: Sanam Shirazi
Student: Yan Shao (Shenia) Tung
August 12, 2019

References

1. Koller, D., Ng, A., Do, C., Chen, Z. (2013). Retention and Intention in Massive Open Online Courses. Educause Review.
2. Davis, D., Chen, G., Jivet, I., Hauff, C., & Houben, G. ([past]). Encouraging Metacognition & Self-Regulation in MOOCs through Increased Learner Feedback. LAL@LAK.
3. Pursel, B.K., Zhang, L., Jablokow, K.W., Choi, G.W., & Velegol, D. ([past]). Understanding MOOC students: motivations and behaviours indicative of MOOC completion. J. Comp. Assisted Learning, 32, 202-217.
4. Macfadyen, L., & Dawson, S. (2010). Mining LMS data to develop an "early warning system" for educators: A proof of concept. Computers & Education, 54, 588-599.
5. Wise, A.F., Speer, J., Marbouti, F., & Hsiao, Y. (2013). Broadening the notion of participation in online discussions: examining patterns in learners' online listening behaviors. Instructional Science, 41, 323-343.
6. Wu, S., Rizoiu, M., & Xie, L. (2017). Beyond Views: Measuring and Predicting Engagement in Online Videos. ICWSM.
7. Bonafini, F., Chae, C., Park, E., & Jablokow, K.W. (2017). How Much Does Student Engagement with Videos and Forums in a MOOC Affect Their Achievement.
8. bigreddot (2018, August 20). Re: How to convert a matplotlib.pyplot to a bokeh plot. Retrieved from https://stackoverflow.com/questions/44423704/how-to-convert-a-matplotlib-pyplot-to-a-bokeh-plot

Stylesheet

1. https://codepen.io/chriddyp/pen/bWLwgP.css

UBC Arts ISIT
Work Learn S19: Learning Analytics/Visual Analytics
Supervisor: Sanam Shirazi
Student: Yan Shao (Shenia) Tung
August 12, 2019

Appendix A – List of Program Files

\README.txt
\appShared.py
\CSVs\Video Quizzes Info\current_QV.xlsx
\CSVs\Video Quizzes Info\past_QV.xlsx
\CSVs\Discussion Posts\current_D.xlsx
\CSVs\Discussion Posts\past_D.xlsx
\CSVs\Discussion Views\current_DV.xlsx
\CSVs\Discussion Views\past_DV.xlsx
\CSVs\Discussion Mapping\current_DM.xlsx
\CSVs\Discussion Mapping\past_DM.xlsx\CSVs\Video Quizzes Info\student_video_quiz[past].sql
\CSVs\Video Quizzes Info\student_video_quiz_current.sql
\CSVs\Video Quizzes Info\student_video_quiz_past.sql
\CSVs\Discussion Posts\posts.sql
\CSVs\Discussion Views\postViews.sql
\CSVs\Discussion Mapping\discussionMapping.sql
\CSVs\Python Outfiles (raw)\rawcurrent.csv
\CSVs\Python Outfiles (raw)\rawpast.csv

UBC Arts ISIT
Work Learn S19: Learning Analytics/Visual Analytics
Supervisor: Sanam Shirazi
Student: Yan Shao (Shenia) Tung
August 12, 2019

Appendix B – Content of Input/output Files

File	Column Names
\CSVs\Video Quizzes Info\current_QV.xlsx \CSVs\Video Quizzes Info\past_QV.xlsx	- course_id, user_id, username, module - numQuizzesAvailable - numQuizzesCompleted - numVidRewatches - numVidUniqueWatched - numVidUnique - quizTimeliness
\CSVs\Discussion Posts\current_D.xlsx \CSVs\Discussion Posts\past_D.xlsx	- course_id, slug_id - slug_type - thread_id - title - username
\CSVs\Discussion Views\current_DV.xlsx \CSVs\Discussion Views\past_DV.xlsx	- course_id, thread_id - username
\CSVs\Discussion Mapping\current_DM.xlsx \CSVs\Discussion Mapping\past_DM.xlsx	- course_id, slug_id - title - module
\CSVs\Python Outfiles (raw)\rawcurrent.csv \CSVs\Python Outfiles (raw)\rawpast.csv	- index - course_id, user_id, username, module - numQuizzesAvailable - numQuizzesCompleted - quizTimeliness - numVidRewatches - numVidUniqueWatched - numVidUnique - numUniqueThreads - numUniquePosts - numPostViewed