MOOCviz 2.0: A Collaborative MOOC Analytics Visualization Platform

Preston Thompson
Kalyan Veeramachaneni
Any Scale Learning for All
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Contents

List of Figures

Abstract

As massive open online courses (MOOCs) continue to grow, the data available to analyze student performance and course effectiveness also grows. Thanks to the standardization of MOOC databases into the MOOCdb format, researchers can write scripts that gather analysis on any MOOC. To provide a platform for sharing these scripts, we have created MOOCviz, a collaborative website for researchers to upload scripts that transform data from a MOOCdb into a HTML-based visualization. Other researchers can view these visualizations and download the scripts to execute on their MOOCs' databases. These researchers can in turn upload the resulting data as a new offering for that visualization, so other researchers and interested parties can compare the same visualization for different MOOCs. MOOCviz 1.0 offered basic functionality for uploading and viewing a new visualization, and was only under development. This paper describes the improvements to MOOCviz 1.0 resulting in the deployed and publicly available MOOCviz 2.0.

First, we added admin approval of new users by customizing one of the ten modules included in the Rails gem Devise and adding a new admin page for approving users. Next, we added the fundamental ability to create new offerings for existing visualizations so that researchers could share the results of previously uploaded visualizations on their own MOOCs. To accommodate multiple offerings per visualization, we added the ability to refresh a visualization when switching between offerings on the visualizations page, which required referencing the parent window from the visualization iframe so we could dynamically change the path to the public data file requested within the iframe. Finally, we added the ability to comment on visualizations to facilitate active discussion among researchers interested in discussing unique observations or improvements to the scripts.

1 Introduction

With the increasing popularity of massive open online courses (MOOCs) has come an increasing amount of data on the demographics and behaviors of the students. The data holds information that can help MOOC researchers learn how to improve MOOCs for future years. Researchers may want to know which resources were used by the top performing students to emphasize those for future students. Researchers might also want to look at stop-out rates and time spent to determine which parts of the course should slow down for more instruction in the future. Researchers can attempt to extract this information by performing various statistical analysis methods on the data collected throughout the duration of MOOCs.

To standardize the storage of this data, the education science MOOC oriented community is adopting the MOOCdb data model developed at MIT [2]. With the shared data model in place, researchers can write scripts that will work not only on their own MOOC data, but on any MOOC data under any platform.

MOOCs are starting to be offered by more and more universities following the lead of platforms like edX and Coursera. Researchers for these platforms spend numerous hours writing comprehensive scripts to *extract*, *aggregate*, and *visualize* the data contained in MOOCdbs. However, researchers have a difficult time collaborating with one another to share both the information they learn as well as which statistical methods have proved most successful for them. Without this collaboration, researchers waste time writing the same scripts as their peers and are not aware of valuable information learned by researchers at other institutions. Additionally, it is hard to compare visualizations and conclusions developed by different researchers when the statistical methods used to create them may have be different and require careful analysis to ensure the same methods and assumptions were followed.

To solve these various problems, MOOCviz was created to be a collaborative website for researchers to share and gather visualizations and analysis methods for MOOCs. Verified researchers can log in to

MOOCviz and upload the data extraction, data aggregation, and data to visualization scripts for a particular visualization they have created. Other researchers interested in a particular visualization can download these scripts and execute the first two scripts their own MOOC databases, coming up with the public data file required for the data to visualization script to create the visualization for their MOOC. Once finished, the researcher can add the offering for their course to the gallery on MOOCviz for others to see and compare with existing offerings. MOOCviz allows anyone to view data visualizations uploaded by researchers for MOOC platforms. The goal of MOOCviz is to provide a collaborative environment where MOOC researchers share their scripts and thoughts on interesting observations on the visualizations or ways to improve the scripts.

2 MOOCviz 1.0

MOOCviz, first proposed in [1], came to fruition through the work of Sherwin Wu, Colin Taylor, and Brian Bell [3] using the Ruby on Rails framework. We will refer to this version as MOOCviz 1.0 in this paper. The following points describe various aspects of MOOCviz when this version was finished, pointing out particular details that will be relevant to discussion of the improvements made in this paper.

Figure 1: Script Pipeline

Before examining the features of MOOCviz, it is important to understand the scripts required for a visualization and how they interact. In the same way that MOOCdb is a standard for the database schema, there is a generalized format for the scripts required to transform the data in a MOOCdb into a MOOCviz visualization. Three scripts are required for each visualization: data extraction, data aggregation, and data to visualization. When the first two scripts are executed on a MOOCdb, they create a public data file that the data to visualization script accesses to create the visualization. This process can be seen in Figure 1. By running the first two scripts on a MOOCdb of a different course, one generates a new public data file from which a new visualization could be acheived. This achieves the goal of generating and visualizing the results on different data sets to share and discuss the results.

State	questions	answers	comments	wiki revisions
A	0.09915744934	0.4335424245	0.420347413	0.04695271324
B	0.1332224878	0.3790917593	0.4543801309	0.03330562195
C	0.1526423225	0.3658425947	0.4690405988	0.01247448401
Else	0.1730502417	0.3870927055	0.3809123397	0.05894471305

Figure 2: A table representation of the .csv public data file for the example visualization in Figure 3.

In detail, the data extraction script queries a MOOC database to retrieve any records that are relevant to the visualization and stores them in the intermediate exported data file, which is not uploaded to MOOCviz.

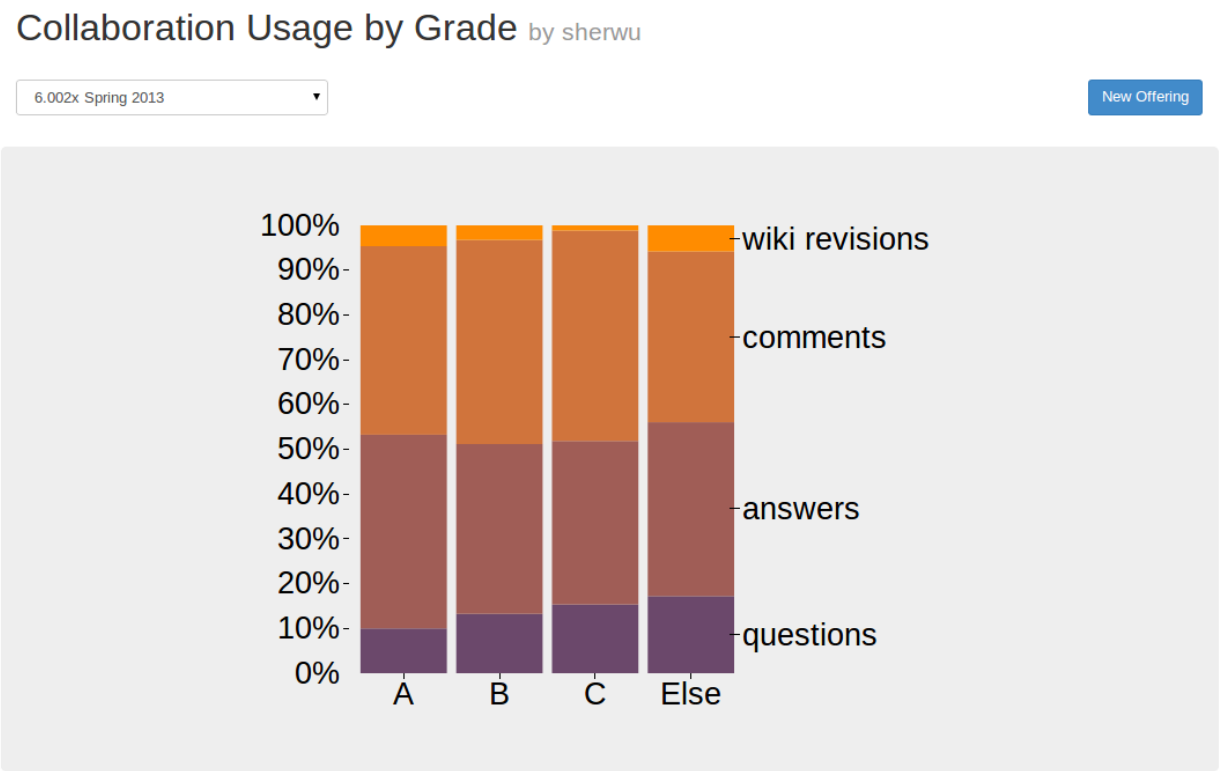

Figure 3: A visualization as it appears on the MOOCviz website.

Next, the data aggregation script combines the exported data into the desired values and stores them in the public data file (example in Figure 2). Finally, the data to visualization script accesses the public data file to create the visualization shown in Figure 3.

This script pipeline lays the foundation for the design of MOOCviz. The salient features of MOOCviz 1.0 were:

- Users were able to sign up for accounts.
- Once logged in, users could go through a process to upload scripts for a visualization and the public data file for a single offering.
- All users, whether logged in or not, could browse through existing visualizations–categorized by user defined tags–and navigate to a page to view a visualization, partially shown in Figure 3. This page showed the visualization and the code of the scripts, and had buttons for users to download the scripts.

Areas of MOOCviz 1.0 that required necessary enhancements were:

- The website was still under development mode and had not been deployed to a server.

- When users signed up for accounts, there was no admin approval, so anyone could create an account and start uploading visualizations.
- There was no option to upload public data files for new offerings of an existing visualization.
- The architecture for storing scripts and public data files was not designed for multiple offerings of the same visualization.

3 MOOCviz Features

For MOOCviz 2.0, we aimed to make the enhancements required after MOOCviz 1.0 and to deploy MOOCviz to a live production server. The subsections below describe the various features we aimed to complete and how we managed to accomplish them.

3.1 Admin Approval of New Users

MOOCviz requires a user account creation and administration interface to accommodate the various roles of MOOCviz users. Any anonymous user may visit MOOCviz without logging in and may view and comment on previously uploaded visualizations. To upload a visualization or create a new offering, however, a user must sign up for MOOCviz and be approved by an administrator. Once a user is approved, they are free to log in and create new visualizations or new offerings by uploading the appropriate files.

To acheive this functionality, MOOCviz uses a Rails gem called Devise[1]. Devise is a customizable 10-module authentication and authorization library that gives programmers the freedom to choose which modules they need and gives many options for tailoring those modules to specific requirements. When a new user signs up, an approval boolean flag is set to false for that user and the admins are emailed indicating there is a new user requesting approval. Admin users have special access to a page listing all users with options to approve any user. When an admin approves a user, sets the approval flag to true. The next time a user tries to log in the user will succeed.

3.2 Enable Multiple Offerings

We would like to give users the option of quickly switching between offerings of a visualization. Therefore, there is a dropdown box at the top of the visualization page listing the names of each offering. When the user selects a new offering, we would like for the visualization to refresh – i.e. have the visualization script execute again – using the public data of the new offering.

When a user navigates to a visualization page, this page must use an iframe to execute the data to visualization script. The data to visualization script at some point references the path to the public data file that it needs to create the visualization. If this reference gives the hardcoded path to the public data file, then when a new offering is selected, the path to the public data file will remain the same. So, we must find a solution other than hardcoding the path to the public data file in the data to visualization script. We have two options:

1. For each offering we can hardcode the path to the public data into a new data to visualization script file stored on the server, and serve the selected offering's data to visualization file to the iframe. Whenever

[1] https://github.com/plataformatec/devise

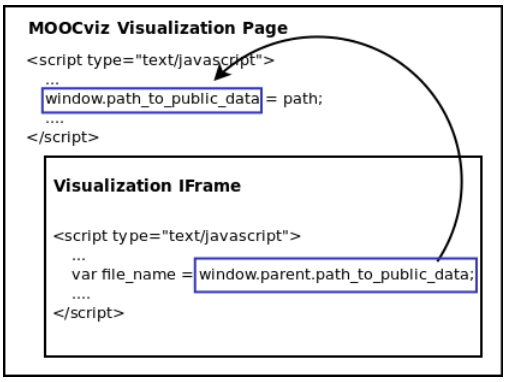

Figure 4: This shows the nested windows of the visualization page. The visualization page window is dynamically created by our Rails backend to assign the correct path to `window.path_to_public_data`, while the data to visualization script iframe is a static file from the server. By using `window.parent.path_to_public_data` in the the iframe, we reference `window.path_to_public_data` in the visualization page window, which we can change via JavaScript before refreshing the iframe if the user selects a new offering.

a user uploads a new offering, we create the new data to visualization file by replacing each path to the first offering's public data file with the new offering's path to the public data file. The benefit of this option is that we do not have to require the user to make any changes to the file before uploading. However, there are several downsides that make this option unattractive:

- We will need to store an additional data to visualization script file for every offering, which scales inefficiently.
- We must create a method to correctly determinine every place the path to the original offering's public data file is used in the originally uploaded data to visualization script so we can replace it each time with new offering's path to the public data file. It would be difficult to ensure that this is method would work under all circumstances.
- Each time a new offering is selected, we would need to change the source of the iframe to that offering's data to visualization script, so we would have to request two new files from the server instead of just one.

2. In the second option, when a new visualization is created, we find every instance of the path to the public data file in the data to visualization script and replace it with code that will access a variable we can change. The iframe's window is the child of the visualization page's window, so the code `window.parent.path_to_public_data` will get the value assigned to `window.path_to_public_data` in the visualization page's window as shown in Figure 4. When a user selects a new offering, we can send an ajax request to the server for the path to the new offering's public data file, assign that value to `window.path_to_public_data`, and then execute the data to visualization script again. This fixes the first and third problems of the first option, although since we still need to replace every instance of the path to the public data file, the first item is still a problem.

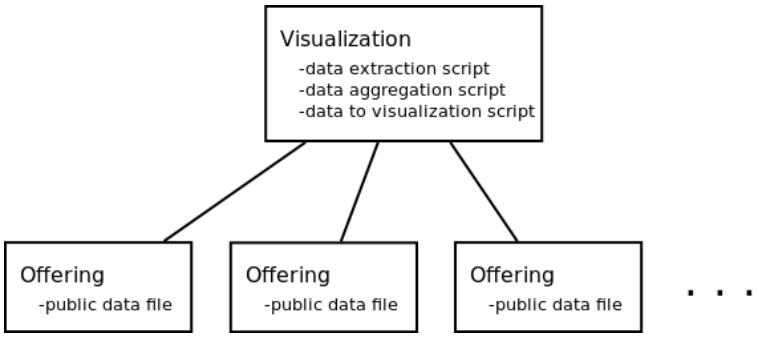

Figure 5: A visualization contains the three scripts, and each offering belonging to that visualization stores its own public data file.

Obviously, it makes sense to choose option two. However, creating a method to replace every instance of the path to the public data file would be tedious and prone to failure. The solution is to ask the user to do this for us, replacing the path with `window.parent.path_to_public_data` before uploading the data to visualization script. This requires more work from the user, but since the users that will be uploading the visualizations will be technical researchers, we assume that they will have the ability to follow instructions for replacing the path to the public data file with the given code, so we included detailed instructions with an example on the website.

3.3 Add a New Offering

MOOCviz allows users to download the scripts to perform the same analysis on a different offering. When a user runs the scripts to create a public data file, the user may go to that visualization page and click the button for a new offering. This will navigate to a new page with a form requesting metadata for the new offering as well as the public data file. Once this information is correctly provided, a new offering is created and MOOCviz allows users to view this offering from the visualization page.

The biggest challenge for uploading scripts is storing them in and accessing them from the file system on the server. We use a Rails gem called Paperclip[2] that allows programmers to treat files as fields of an object model. MOOCviz 1.0 stored the visualization scripts and the public data file as fields of a model called uploads, which belonged to an offering, which belonged to a visualization. This representation fails to recognize that the scripts are the same for each offering of a visualization and therefore should belong to a visualization and not an offering. The public data file, however, changes with each offering, so that file should belong to an offering. To fix this, we removed the upload model, added the three scripts as three distinct fields for a visualization, and added the public data file as a field for an offering as shown in Figure 5. With this representation in place, adding a new offering simply requires setting the public data file field to the newly uploaded public data file. To access the scripts for this offering, you simply need to get the offering's visualization, then get the scripts stored as fields for that visualization.

[2]https://github.com/thoughtbot/paperclip

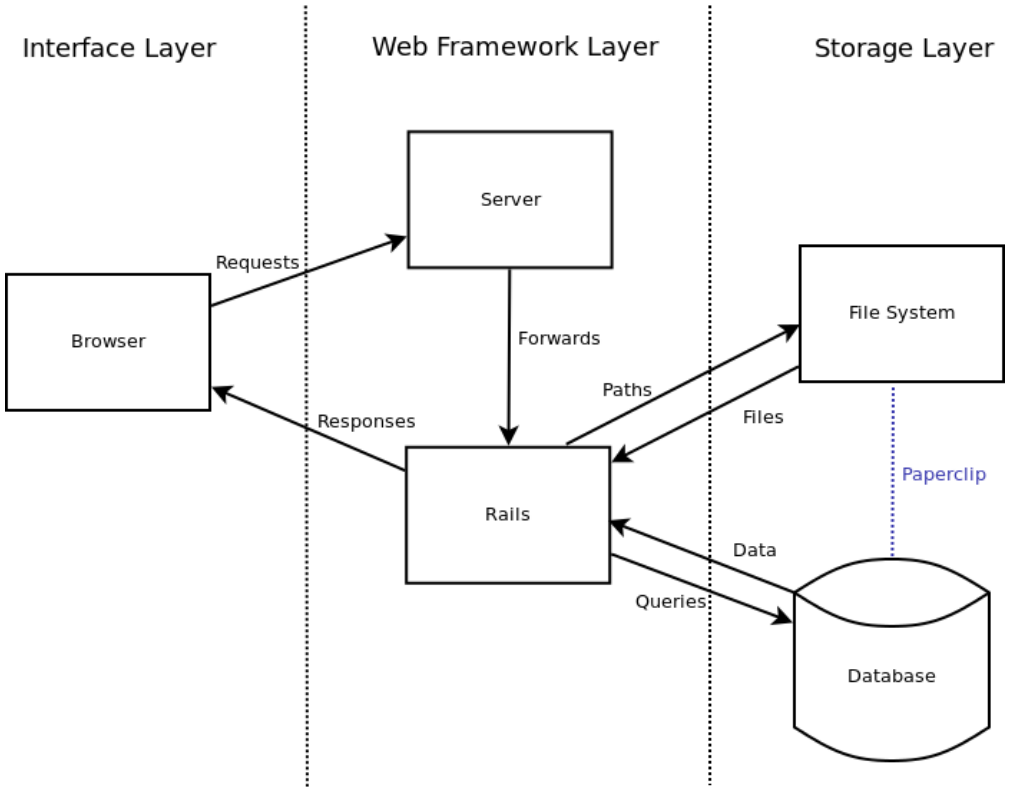

Figure 6: MOOCviz Architecture

3.4 Comment on Visualizations

With the hopes of facilitating active discussion about visualizations, we incorporate a section of the visualization page to comment on the visualization. If a user is logged in when a comment is posted, their username will appear next to that comment; otherwise, the comment will remain anonymous. For further versions of MOOCviz, we designed these comments to easily extend to be about other parts of the website (e.g. Side-by-Side Comparisons in Section 5.1).

4 Architecture

This section describes the MOOCviz architecture diagram in Figure 6, paying particular attention to the Storage Layer since it contains most of the architecture specific to MOOCviz. A browser sends requests to a server in the form of URLs, which the server forwards to Ruby on Rails. Rails determines which action it should take based on the various properties of the request. Most times, the actions involve querying the database for information or to update information about users, visualizations, offerings, and/or any other models or relationships between them. The database schema for MOOCviz is detailed in Figure 7. These

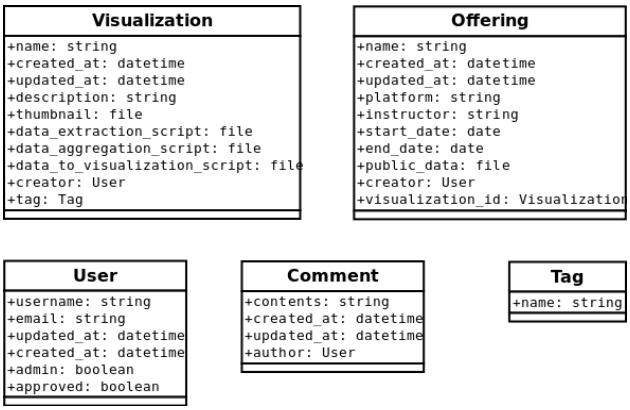

Figure 7: MOOCviz Database Schema

actions also often involve accessing the server's file system to find static files. Paperclip is an interesting gem in that it stores information about files in the database, and uses queries to figure out where in the file system to find the files attached to models. When Rails has finished performing the action, it returns a response to the browser.

5 Future Work

5.1 Compare Offerings Side-by-Side

One of the main reasons MOOCviz was created was to allow the same analysis on multiple MOOCdbs to achieve comparable results. To facilitate this, one area of future work is to create a page where users can select two offerings of the same visualization to view simultaneously. The purpose of this page is to provie a quick and easy way to make acute observations about the differences. This page will also feature a comments section so users can share their observations about these comparisons. These comments may simply be the same comments from the visualizations page, but a more interesting possibility is to create a unique comment thread for each comparison, allowing a more tailored discussion about a specific comparison.

5.2 Filter Visualizations

As of now, MOOCviz's homepage features a collection of thumbnails linking to the visualization pages. The thumbnails are sorted by the visualization's tag. However, as the number of visualizations grow, it will become impractical to list all of the visualizations on the homepage, and will make it difficult for users to find a particular visualization or discover new visualizations. To fix this, a future area of work will be to provide options for the user to sort, filter, and search visualizations by various factors. Some basic filters could be recently added, recently commented on, and most viewed. Other more advanced filters could allow users to filter the visualizations based on their metadata like which platform offered the course, or during what timeframe the course was offered. These filters could even be used to find specific offerings within

visualizations. With these filters in place, users will have a much better experience finding visuallizations that interest them, leading to higher engagement and an overall more active MOOCviz community.

6 Conclusion

MOOCviz was created for MOOC researchers working under different platforms to collaborate with one another by sharing scripts through the MOOCviz framework. By sharing the MOOCdb data standards and the script pipeline, researches can upload their scripts to MOOCviz. Then, other researchers can download the scripts, run them on their offering's data, and upload the results back to MOOCviz. This creates an environment when researchers can compare the same visualizations on different data sets to discuss the merits of different ideas within the MOOC community and to further the pursuit of better MOOC strategies in the future.

References

[1] Franck Dernoncourt, Colin Taylor, Una-May O'Reilly, Kalyan Veeramachaneni, Sherwin Wu, Chuong Do, and Sherif Halawa. MoocViz: A Large Scale, Open Access, Collaborative, Data Analytics Platform for MOOCs.

[2] Kalyan Veeramachaneni, Franck Dernoncourt, Colin Taylor, Zachary Pardos, and Una-May O'Reilly. MOOCdb: Developing Data Standards for MOOC Data Science. In *AIED 2013 Workshops Proceedings Volume*, page 17, 2013.

[3] Sherwin Wu. MOOCdb: An collaborative environment for MOOC data. Massachusetts Institute of Technology 6.UAP Final Report, December 2013.