May 2019

Fair Ranking System

Paul-Henry Madiba Schoenhagen
Worcester Polytechnic Institute

Creating A Fair Ranking System

A Major Qualifying Project

Submitted to the Faculty of

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree in Bachelor of Science

in

Computer Science

By

Paul-Henry Schoenhagen (CS)

Submission Date:

5/30/2019

Project Advisor:

Professor Elke Rundensteiner

Acknowledgements

Major thanks to Caitlin Kuhlman, Elke Rundensteiner, Lane Harrison.

Abstract

Ranking greatly assists us in decision making by allowing us to consider numerous options, with a variety of attributes and turn it into an understandable model. Yet, accurate rankings can be difficult for one to construct without extensive and proper knowledge on what you’re ranking. Thus arose the need for fair ranking systems that effectively elicit data from users and produce accurate rankings that satisfy and serve the users’ needs. In the pursuit of creating a fair ranking system this team proposes Rankit_Experimenter, a user study that tests the merits of three preference collection methods. Results indicate that categorical binning is the best value model: providing a substantial amount of data for the underlying ranking algorithm, while requiring minimal effort from the user.

Table of Contents

List of Figures

Executive Summary

People depend on rankings created by companies and institutions for assistance in everyday decisions; anywhere from getting show recommendations on a streaming service to deciding which search engine link to click. Unfortunately these rankings are not designed with an individual person in mind, companies optimize for their own objectives and business goals. Therefore if an individual wishes to rank a set of options according to their preferences they must use an interactive ranking system. The goal of this project is to create a refined build of the RanKit system to test key features and gather quantitative user data. RanKit is an online ranking application previously developed by a team of WPI students. With this goal in mind, we reduced RanKit to its core feedback loop, and focused on a few key features we wanted to test. Although most user studies are done to test the intuitiveness of an application, this was done to collect quantitative user interaction data. Using a framework built for gathering user data and conducting extensive user studies, the build was evaluated and shown to successfully accomplish the aforementioned goal.

Rankit_Experimentr, the product of this project, marries RanKit with Experimentr: a framework built by WPI professor Lane Harrison designed for collecting user data on large scale user studies via Amazon's Mechanical Turk. Here RanKit was condensed into: a page for the user's consent to participate, an instructions page, a page for entering partial rankings, a page to explore the complete ranking, a page to answer predetermined questions, and a page for additional feedback. Two different types of user studies were carried out to gather sufficient user data and test the stability of the build: in-person and online. The in-person study was carried out on a smaller group, consisting of college students and librarians, whilst the online testing was carried out by vetted volunteers.

Chapter 1: Introduction

Everyday decision making depends on the ability to process a large amount of data and constructing a simple model for evaluation. These models are meant to provide us with more information on each option, the attributes associated with it, and ultimately assist us with decision making. Ranking, a commonly used model, processes the information and arranges the options in an ordered list. This allows an individual to determine the relative weight of each option, and make an informed decision.

Manually ranking a large dataset by oneself proves to be a troublesome task. Attributes can be missed when turning real world events and objects into data, and outliers can greatly affect the weights of different attributes. Even after the cleaning and synthesizing of data, the data might be too large to analyze in a reasonable amount of time, too incomplete to make an accurate ranking, or too biased to the the point where, within a vacuum, the ranking would be misleading. Ranking a large dataset becomes increasingly more difficult for an individual without extensive time, knowledge of what they are ranking, or the correct tools.

Thus people depend on rankings created by companies and institutions for assistance in making decisions; such as getting show recommendations from a streaming service, determining where to eat for the night, or deciding which search engine link to click. These rankings are designed with the intention of appealing to a large demographic rather than appealing to an individuals preferences. And without any knowledge of the ranking algorithms these systems use, this leads to users feeling like they "fighting the model", [4]. Where the users were unsatisfied with rankings since it felt like their opinions are not sufficiently represented. A tool that processes a large datasets, represents users preferences, and provides an accurate ranking would prove to be more valuable than a generic ranking.

With that in mind a previous team of WPI students build RanKit, an online tool that gives users access to learning-to-rank algorithms. Users are encouraged to create a partial ranking of objects within a dataset; based off that input the system then constructs a global ranking.

Building a ranking off users' preferences does two things: it makes the process of ranking a large dataset much easier, and it builds trust between the user and the system.

The objective of this project was to further develop RanKit as a fair ranking system. This was done by redesigning RanKit into a build specifically test key features, running multiple user studies, and studying user behaviour data. The redesign of RanKit was done using Experimentr; a framework developed by WPI professor Lane Harrison. The online user studies were carried out on the Amazon Web Service Mechanical Turk. This redesign is not permanent, and only exists to be a user study meant to capture data on user behaviour.

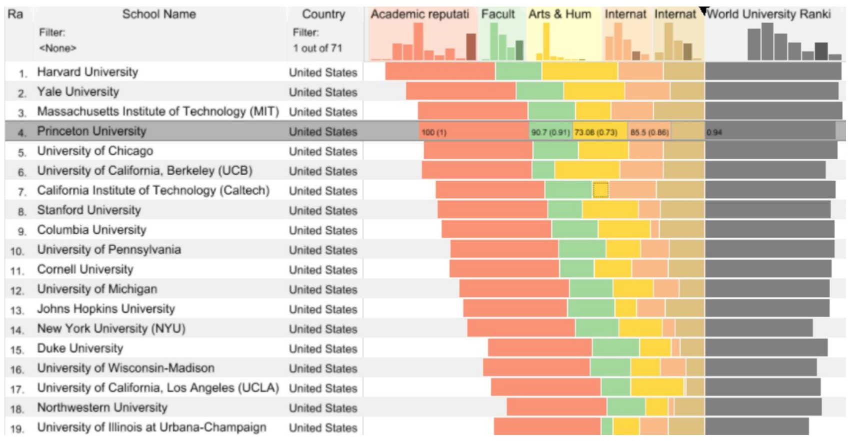
Chapter 2: Background

2.1 Related Work

2.1.1 LineUp

LineUp is a visualization system that uses bar charts to display data. It supports "the ranking of items based on multiple heterogeneous attributes with different scales and semantics." [7]**.** LineUp allows for the direct manipulation of attributes, in multi-attribute items, as well as enabling users to change the weights associated with each attribute. The bar charts are used to represent the weights of each attribute, this allows users to easily notice the impact of their choices in real-time. The integration of slope graphs can be used to compare alternate rankings of the same dataset.

The ability to adjust the weights and attributes inspired RanKit to have user preferences be the driving factor when constructing rankings. The difference between RanKit and LineUp is that RanKit must derive the weights of the attributes based off the user's partial rankings.



Ra	School Name	Country	Academic reputati	Facult	Arts & Hum	Internat	Internat	World University Ranki
	Filter: <None>	Filter: 1 out of 71						
1.	Harvard University	United States						
2.	Yale University	United States						
3.	Massachusetts Institute of Technology (MIT)	United States						
4.	Princeton University	United States	100 (1)	90.7 (0.91)	73.08 (0.73)	85.5 (0.86)		0.94
5.	University of Chicago	United States						
6.	University of California, Berkeley (UCB)	United States						
7.	California Institute of Technology (Caltech)	United States						
8.	Stanford University	United States						
9.	Columbia University	United States						
10.	University of Pennsylvania	United States						
11.	Cornell University	United States						
12.	University of Michigan	United States						
13.	Johns Hopkins University	United States						
14.	New York University (NYU)	United States						
15.	Duke University	United States						
16.	University of Wisconsin-Madison	United States						
17.	University of California, Los Angeles (UCLA)	United States						
18.	Northwestern University	United States						
19.	University of Illinois at Urbana-Champaign	United States						

Fig 1. Lineup interface

2.1.2 Podium

Podium is another ranking system that factors users' preferences into the final ranking. Podium allows users to users to drag rows in the table "to rank order data points based on their perception of the relative value of the data" [4]. The system then creates a weighting model that best fits the users preferences. The inclusion of an interactive table allows the system to infer the best weighting model and helps users to easily visualize the rankings they are creating.

Unfortunately the rankings Podium provided were not the most accurate due to users not spending short amounts of time with the tool, thus giving the system very little to infer what a users' preferences are. RanKit differentiates itself from Podium by having three modes of comparison, opposed to Podiums single mode, as well as including minor feature, called *Motivators,* that encourage the user to input a sufficient amount of data.

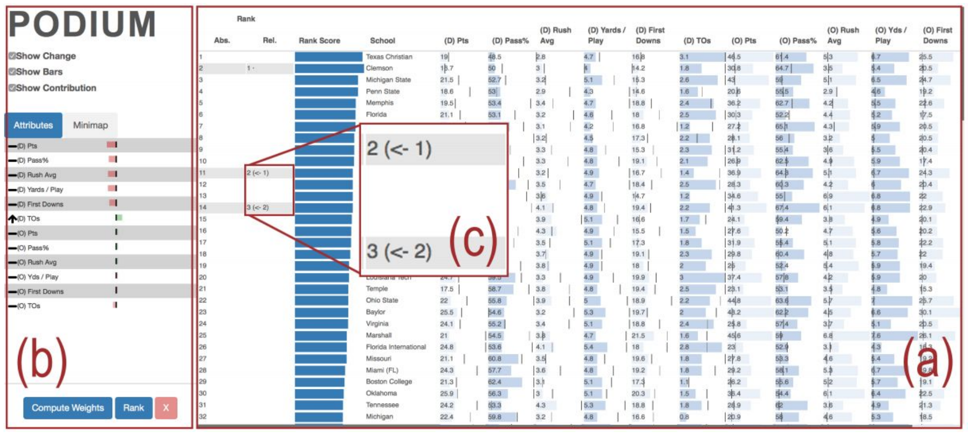
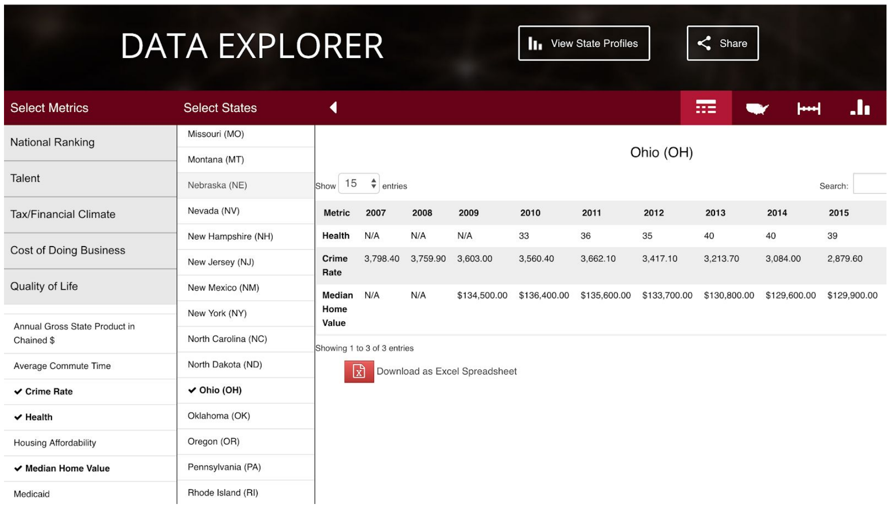


Fig 2. Podium interface

2.1.3 MATTERS

MATTERS, developed in part by WPI, stands for Massachusetts Technology, Talent, and Economic Reporting System. It is a tool used to compare different states with published data aggregated from a variety of different sources [16]. The tool provides data visualizations so users can easily rank states based on attributes related to talent, taxes, business, and quality of life. The four different types of visualizations MATTERS offers are the: table view, heat maps, line charts, and bar charts.



Ohio (OH)

Metric	2007	2008	2009	2010	2011	2012	2013	2014	2015
Health	N/A	N/A	N/A	33	36	35	40	40	39
Crime Rate	3,798.40	3,759.90	3,603.00	3,560.40	3,662.10	3,417.10	3,213.70	3,084.00	2,879.60
Median Home Value	N/A	N/A	$134,500.00	$136,400.00	$135,600.00	$133,700.00	$130,800.00	$129,600.00	$129,900.00

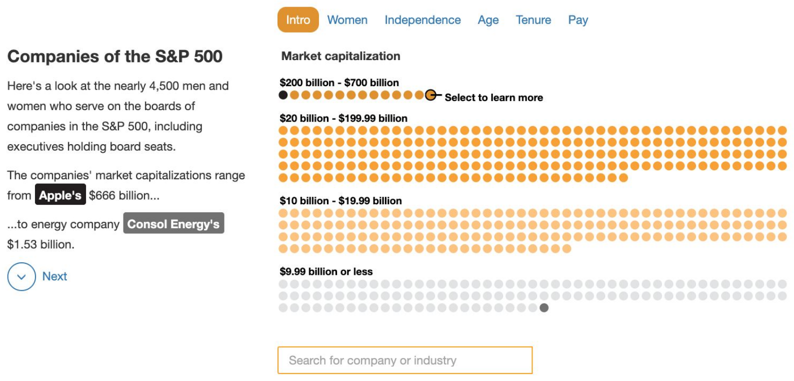
Showing 1 to 3 of 3 entries

Fig 3. MATTERS interface

MATTERS providing visualizations to better explore the final ranking was a design choice that was later adopted by RanKit.

2.1.4 Effects of Adding Search Functionality to Interactive Visualizations on the Web

Led by Lane Harrison, this team of WPI and Bucknell students created a user study that employs the Experimentr.js framework. The purpose of this experiment was to study user behaviour when a web visualization implements text-based search. The project's focus on user behaviour and integration with Experimentr.js made this an invaluable resource to familiarize oneself with [6].


Intro
Women
Independence
Age
Tenure
Pay
Market capitalization
$200 billion - $700 billion
Select to learn more
$20 billion - $199.99 billion
$10 billion - $19.99 billion
$9.99 billion or less
Next
Search for company or industry

Companies of the S&P 500

Here's a look at the nearly 4,500 men and women who serve on the boards of companies in the S&P 500, including executives holding board seats.

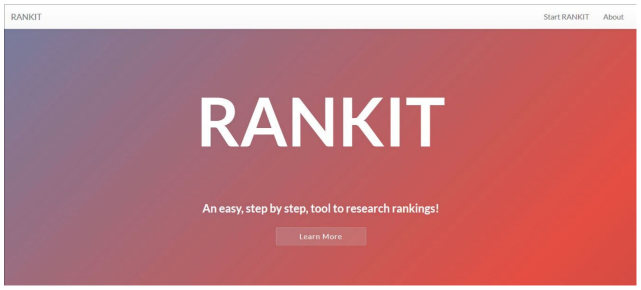
The companies' market capitalizations range from Apple's $666 billion...

...to energy company Consol Energy's $1.53 billion.

fig 4. Search-in-vis Board of Directors Experiment Interface

2.2 RANKIT

2.2.1 RanKit Overview



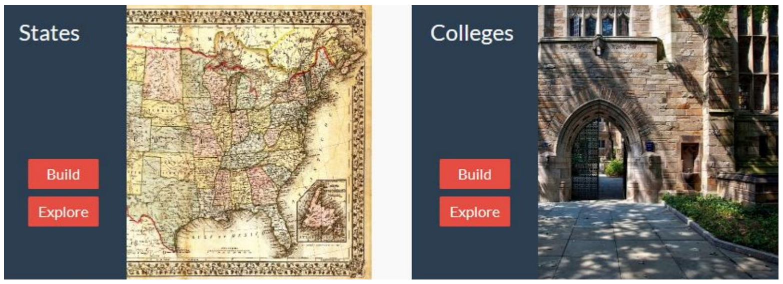


fig 5.1 (left) RanKit Landing Page, fig 5.2 (right) Dataset Choices

RanKit is a web application, with its key feature being the Build Tool: a page that offers multiple modes of comparison. The general flow is that you start on the landing page, select which dataset you're interested in ranking, and finally select which mode of comparison you'll be using to make your ranking. The three possible modes are List, Categorical, and Pairwise comparison. You then create your partial ranking, and the system will take you to the Explore page so you can interact with the final ranking.

2.2.2 RanKit Methodology

RanKit is " an online ranking application that provides solutions for analyzing and exploring rankings. The system uses machine learning to automatically construct rankings based on partial input from users." [3]. Rankit employs Javascript on the client-side, and Python on the server-side. This setup allowed the developers to use efficient web frameworks, provides external libraries for the communication between frontend and backend servers, and make heavy computations on the server-side without it greatly affecting the application performance. The use of the web framework Flask was used to transfer JSON-formatted data between the Javascript client and the Python server. The client was built to be innovative and intuitive, following modern design practices for web applications. Twitter Bootstrap Cascading Style Sheets were used so that the tool was responsive and that it's styling was consistent. Python is a high-level object-oriented programming language, having libraries specifically for machine learning, and tools for data analysis. Python has low latency and trouble communicating with the frontend server, an issue that becomes apparent when dealing with large datasets.

Three different JSON description files are used for communication between the client and the server. The first loads the dataset into the Build tool, the second sends the users preferences to the Build machine learning script, and the last sends the global ranking to the Explore tool. In order for data to persist between page loads, the URL was manipulated to hold data about the users ranking.

Despite many different ways to do machine learning scripts for ranking, RanKit continued to build of MyRanker [12]. RanKit uses the RankRLS library that uses an altered version of the regularized-least squares algorithm [14].

The key feature of RanKit is the build page. The build page consists of the items of the dataset on the left side of the screen, and the rankable boxes on the right. The user has the ability to search for specific items, sort the dataset alphabetically, and shuffle the dataset into a random order. The Build page offers three comparison models for users to input partial rankings: List, Categorical, and Pairwise. Partial ranking from List and Categorical Comparison are converted to Pairwise rankings because the ranking script only works when the data is formatted this way.

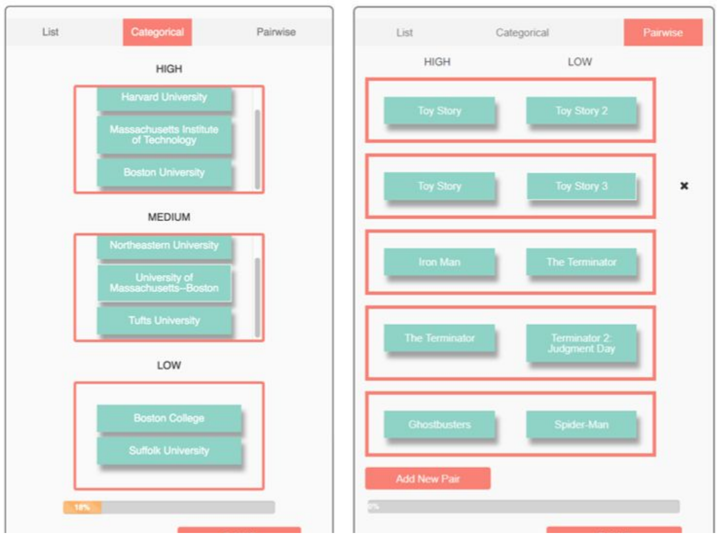


fig 6. Categorical and Pairwise Comparisons

In the case of a List comparison, where items are ranked in descending preference, a list of A, B, and C is understood as:

- A is preferred over B
- A is preferred over C
- B is preferred over C

Therefore for [O_1 , O_2 , … , O_n] where O_i is the ith item in the list and n > 0, there are (n2 -n)/2 number of pairs generated: reducing the amount user data needed for a Pairwise Comparison method by an order of 2.

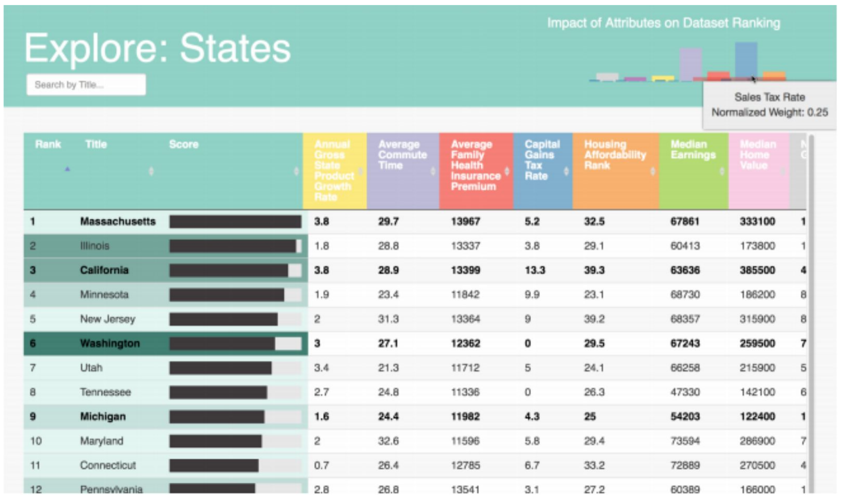
In the case of a Categorical Comparison users place items into either High, Medium, or Low brackets, these brackets correspond to the users preferences. Unlike List comparison, the order of items within the brackets does not matter. If a user were to place A and B into the high preference bracket; C and D into the medium preference bracket; and F into the low preference bracket. It is understood as:

- A is preferred over C
- A is preferred over D
- A is preferred over F
- B is preferred over C
- B is preferred over D

- B is preferred over F
- C is preferred over F
- D is preferred over F

With Categorical comparison users input partial ranking over a set of items {O1 , O2 , … , On} but not for every (Oi , Oi+1) pair. Meaning Categorical comparison is not as concise and easy as List comparison, but is more flexible like Pairwise comparison [3]. Users are also offered the ability to explicitly create these pairs with Pairwise comparison. This method gives full control to the users at the expense of the ease and conciseness of the other comparison models.

The Explore page displays the final ranking in a table format. Users were initially having problems understanding the table so the ability to view the weights of each attribute was added in a later build.



Rank	Title	Score	Annual Gross State Product Growth Rate	Average Commute Time	Average Family Health Insurance Premium	Capital Gains Tax Rate	Housing Affordability Rank	Median Earnings	Median Home Value	N G
1	**Massachusetts**		**3.8**	**29.7**	**13967**	**5.2**	**32.5**	**67861**	**333100**	**1**
2	Illinois		1.8	28.8	13337	3.8	29.1	60413	173800	1
3	**California**		**3.8**	**28.9**	**13399**	**13.3**	**39.3**	**63636**	**385500**	**4**
4	Minnesota		1.9	23.4	11842	9.9	23.1	68730	186200	8
5	New Jersey		2	31.3	13364	9	39.2	68357	315900	8
6	**Washington**		**3**	**27.1**	**12362**	**0**	**29.5**	**67243**	**259500**	**7**
7	Utah		3.4	21.3	11712	5	24.1	66258	215900	5
8	Tennessee		2.7	24.8	11336	0	26.3	47330	142100	6
9	**Michigan**		**1.6**	**24.4**	**11982**	**4.3**	**25**	**54203**	**122400**	**1**
10	Maryland		2	32.6	11596	5.8	29.4	73594	286900	7
11	Connecticut		0.7	26.4	12785	6.7	33.2	72889	270500	4
12	Pennsylvania		2.8	26.8	13541	3.1	27.2	60389	166000	1

fig 7. Explore Page

Chapter 3: Methodology

3.1 Overview

Before work could be started on RankIt_Experimentr, I needed to build a familiarity with the pre-existing project and secure the ability to conduct a user study. This meant reading previous work on the subject, interacting with the site, refactoring code, and fixing bugs. This was a necessary step in the development process since I gained a real understanding of how the project worked, which proved to be helpful when integrating RanKit with Experimentr.js, which I will refer to as Experimentr.

3.1.1 Early Work

Code was refactored so the JavaScript no longer ran directly from the webpage. Next Datatables, which is a plugin for the jQuery library, was integrated to fix the problems we were having with the table being displayed on the Rank Exploration page. By using Datatables I was able to fix the header and first three columns in place, making valuable information still visible while the user was scrolling either horizontally or vertically on the table.

Popover data was then added to items in the final ranking, displaying the system's confidence in the ranking as well as the overall score of each ranked item. These popovers would display when the item was clicked, and would disappear when the user would click elsewhere on the page. The final fix was associated with what were called "motivators" i.e the confidence bar underneath the users partial ranking and the barcharts displayed in the top right corner of the page. Due to the process in which it was originally rendered, reloading the page while ranking items would cause the motivators to first vanish, and then display inconsistent percentages before and after the page reloaded.

3.1.2 Institutional Reform Board (IRB) Specifications

The Institutional Reform Board is an independent ethics committee that abides by FDA regulations, their purpose is to review any biomedical or behavioural study that involves human subjects. They are able to either accept, deny, or request changes to any presented study, on the basis of whether or not the study is ethical [9].

Since the many aspects of our user study was different than the original Rankit user study; an updated IRB form was necessary to address the changes in the nature of the experiment, the people working on the study, and other details about the study. A digital copy of the IRB form was included within the final experiment for each participant to read and consent to.

3.2 Redesigning RanKit For Experimentr

This project is a mixture of Experimentr and RanKit. Developed by WPI professor Lane Harrison, Experimentr is a front-end framework built with Node.js, Redis, and D3.js, amongst a few other frameworks. The framework is primarily used for web-based visualization studies, where each stage of the study is broken up into *Modules*; which consist of HTML and Javascript. We chose Experimentr as the framework to build the user study out of because of the data collection capabilities it offered, the tools and focus it had towards conducting user studies, and the ability to communicate with the head developer Professor Lane Harrison [13].

3.2.1 RanKit Modifications

RanKit, as an online tool, had previously been tested by regular users and UI experts on its usability with metrics such as usability, intuitiveness and qualitative features. Whereas this was meaningful data, our focus was to get quantitative data on the way users interacted with the core feedback loop. The loop being: creating a partial ranking of a dataset, exploring the global ranking, and being able to go back adjust their partial ranking. This meant some features and sections of the Rankit website either had to be stripped or condensed.

- **Motivators -** The miniature barchart in the top right corner representing the weights the ranking algorithm gives to each attribute, as well as the live confidence bar that provided instant feedback were removed from the final product, due to early issues we had with calculating the confidence and weight values of ranked items that came from integrating RankIt with Experimentr.

- **Dataset Options -** Unlike Rankit, which offers a myriad of datasets to rank from, we sought a single dataset for this study. Since we didn't know what the demographic of our user study would be, the database had to be a topic that an average user would have knowledge about. After narrowing it down between the college and movie dataset, we also had to consider which dataset would be best for the algorithm. Meaning: which dataset had attributes that the algorithm could use to efficiently create an accurate ranking. After some deliberation we decided that the college dataset would be the best to go with due to its commonality and quantitative attributes that the algorithm would have an easier time ranking.

- **Choice of Comparison Method -** Previous work found that users overwhelmingly preferred Categorical comparison over both List and Pairwise comparison. Therefore in an effort to test all methods evenly and to keep all interactions within the experiment focused on ranking; we removed the ability for participants to choose which comparison method to use, opting instead to assign each participant a random method of comparison.

- **Additional Pages -** The dataset page was stripped since participants no longer were able to choose which dataset to rank. The "About" page was stripped due to much of the information on what the tool is and how to operate it being distributed amongst other modules. Other additional pages on the myRanker website were stripped due it no longer serving a purpose to our user study.

3.2.2 Early Designs

This condensed version of Rankit was then fashioned to operate as an online user study that works within the bounds of Experimentr. Mentioned before, Experimentr is a front-end framework that was created to carry out user studies, with each step of the user study being separated into its own module. Figure 8 is the initial design as to how the online user study would be iterated through: with the larger boxes representing each separate module, and the smaller boxes representing the action needed to take you to it.

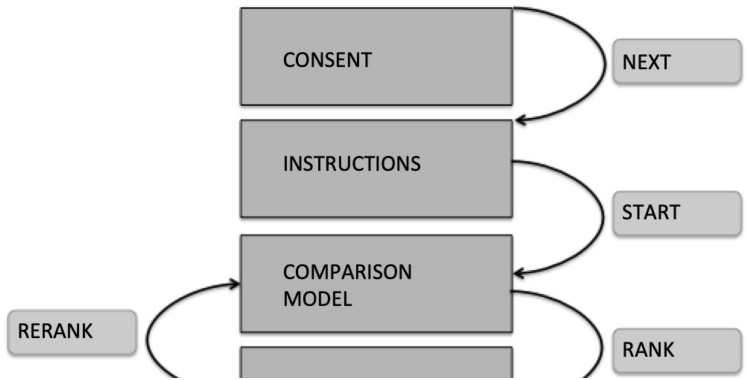


fig 8. Early Rankit_Experimentr Iteration Design

In the end another module, Strategy, was added between the Explore and Debrief module. This module offered a more focused questionnaire centered around the users interactions with the tool itself, with the option to leave more general feedback remaining in the Debrief module.

3.3 Design And Implementation

3.3.1 The Server

The structure of our backend was integral to project success, due to the amount of unseen tasks the project had to complete. Our backend consists of Python, Node.js and Redis.

Python can be a great language for backends. It is a mature, interactive programming language that has extensive library support, built-in tools for data analysis, and many advanced web APIs. Amongst the libraries available are those for machine learning, which is necessary since our tool uses machine learning to create the final rankings. Additionally, Python can complete extensive server-side calculations without majorly hindering the applications performance. However some developers have begun to shy away from it due to its low latency and slow performance [10].

Node.js is commonly used and is widely popular amongst the web developer community. Node.js offers Node Package Manager (NPM) which makes installing third party packages, dependency management, and version management much easier. It helps avoid version conflict by allowing you to isolate packages from other projects. In addition, It is a JavaScript runtime built using Google's V8 engine for use in Google Chrome. V8 is a fast and efficient engine because it compiles JavaScript into native machine code. Since Node.js is written in JavaScript, it would result in most of project being written in one language [1].

Redis is an open source, in-memory data structure store that was downloaded with the use of NPM. Primarily used as a database, it can also function as a cache and message broker. Redis supports many different data structures, including but not limited to: strings, lists, bitmaps, hashes and sets. Redis is able to make reads much faster because it loads data onto RAM, it also provides the ability to periodically write data onto disk, allowing data to be persistent. Overall Redis is a lightweight data store that allows for fast storage and retrieval of data. Experimentr requires Redis in order to operate, but considering the benefits it provides it's apparent why it was chosen for their web-server needs [8].

In totality the server uses a hybrid of JavaScript for client-side interactions, Python as the server-side language needed for machine-learning and ranking, and Redis for server-side data collection. This structure allows for web frameworks and external libraries that set up communication between the JavaScript client and both the Python and Redis server, while also keeping data sent to each respective server isolated.

One such framework used was Flask, which eased the process of passing JSON objects between the servers and client. Jinja2 was specifically used for interactions between the client and Python server [3]. Python is necessary for RankIt, due to it being able to do heavy calculations and providing built-in machine learning libraries, yet there was still a considerable amount of time spent mitigating Pythons slow communication with the server when handling a larger amount of data.

3.3.2 Machine Learning Algorithm

RanKit's ranking algorithm that we started with, and the one that was later switched to both followed the pairwise approach of ranking objects; which turns the ranking problem into a pairwise classification or a pairwise regression. This allows a ranking to be learned by applying a binary classifier to order pairs of objects.

RanKit's machine learning script utilizes the RankRLS library, which uses a variation of the regularized-least squares algorithm called RankRLS [14]. The algorithm reads a list of pairs and a label informing the algorithm which object in a pair is preferred over the other. The pairs contain the partial ranking data provided by users. The pairs are training data that the algorithm needs to formulate a model, which is then used to predict a final ranking over the entire dataset.

For later builds of Rankit_Experimentr The RankSVM algorithm was used. RankSVM [19] uses a Support Vector Machine (SVM) algorithm to distinguish between correctly ordered and incorrect pairs of objects. Therefore, instead of learning the ranking from each individual object in the training dataset, the difference between pairs of objects is measured and that difference becomes a vector. The calculated vectors will then be used to create a combined feature vector which the function will be learned from. Each training pair has a label which indicates where a pair is correctly ordered, or inverted. The weight vector is a hyperplane decision boundary which distinguishes between these two classes while maximizing the space between them. Once the boundary has been learned from the training data, a global ranking including the rest of the dataset can be created. An object's final rank position is determined by a score given to it by the algorithm.

One disadvantage we faced compared to a usual supervised learning problem formulation, is that with a supervised learning problem the true ranking the dataset is known. For interactive ranking, the problem is semi-supervised [15], in that labels are given only to a portion smaller than the entire dataset.

It's important to note the amount of training pairs needed to effectively create a model that provides a satisfactory ranking. It has been observed that if pairs are selected at random and labeled, then the RankSVM algorithm needs O(n) pairs to perform optimally and produce results that are better than randomly expected [5].

3.3.3 The Client

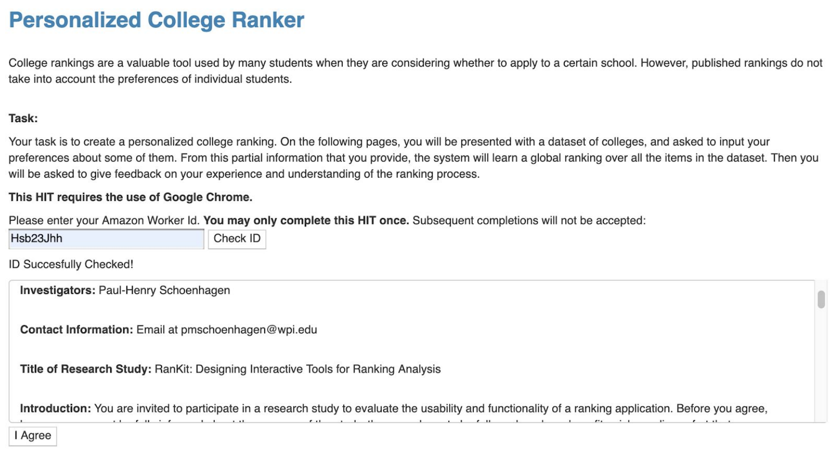
Providing a straightforward and simple user interface was the focus of the design throughout the project. Partially to keep the participant focused on the functionality and intuitiveness of the tool as opposed to its appearance, mostly because we were planning an online user study and wanted it to be easily accessible to the widest demographic possible. Although initial module designs stemmed from the templates Experimentr offered, changes were made throughout the development cycle in response to feedback.

When using Experimentr, each step of the study is separated into modules. Modules contain HTML and javascript and are directly loaded into a solitary webpage. Since no new webpages are created the participant cannot iterate through modules using their web browsers' forward and back buttons. This ensures participants interact with the experiment in its intended manner. As previously mentioned, to compensate for the inability to choose which comparison method to use, Rankit_Experimentr randomly assigns a method to each participant.

All figures shown in the following section were captured from a test run, as to not compromise any personal data gathered from our participants.

Consent Module

The first module users interact with is the consent module, with that in mind it was designed to provide the participant with more information on the study. The first thing seen is the title. We decided to use “Personalized College Ranker” for the title instead of “Rankit” or “Rankit_Experimentr” because it is a more focused title that immediately informs the participant what they'll be doing within the study. The next section describes the purpose and task they’ll be doing. Due to Experimentr’s designed integration with Mechanical Turk, the next section asks for their Amazon Worker ID and gives a reminder that Google Chrome is needed.



Personalized College Ranker

College rankings are a valuable tool used by many students when they are considering whether to apply to a certain school. However, published rankings do not take into account the preferences of individual students.

Task:

Your task is to create a personalized college ranking. On the following pages, you will be presented with a dataset of colleges, and asked to input your preferences about some of them. From this partial information that you provide, the system will learn a global ranking over all the items in the dataset. Then you will be asked to give feedback on your experience and understanding of the ranking process.

This HIT requires the use of Google Chrome.

Please enter your Amazon Worker Id. **You may only complete this HIT once.** Subsequent completions will not be accepted:

Hsb23Jhh

ID Succesfully Checked!

Investigators: Paul-Henry Schoenhagen

Contact Information: Email at pmschoenhagen@wpi.edu

Title of Research Study: RanKit: Designing Interactive Tools for Ranking Analysis

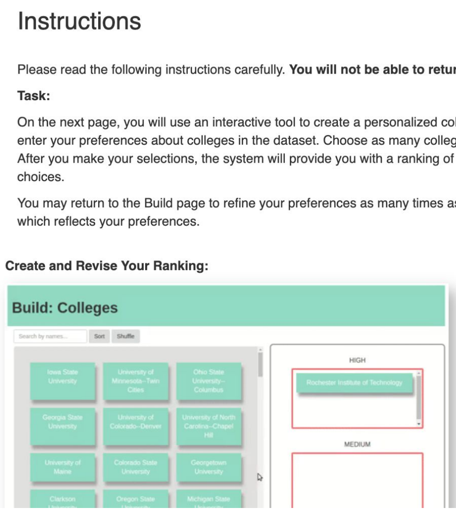
Introduction: You are invited to participate in a research study to evaluate the usability and functionality of a ranking application. Before you agree,

fig 9. Consent Module

The last thing is a digital version of the Informed Consent Agreement for Participation in a Research Study, which can be viewed in *Appendix A*, this includes in-depth information about the study and the rights afforded to them as a participant. In order to progress past this page participants must click “I Agree”. By doing this they agree that they have been properly informed about the study, and consent to taking it.

Instructions Module

After consenting to to take the study participants are brought to the Instructions module which has a more descriptive set of instructions pertaining to which comparison method they were randomly assigned. Each set of instructions followed the same formatting: an introductory paragraph restating what to expect of the experiment, descriptions on how to rank and interact with the data, and a notice that there will be questions to answer at the end. The script for the full set of instructions was written to inform the participant of the tasks that they'd be doing as well as in what order, the training script can be read in *Appendix C*. In order to progress participants must click "Start Exploring >>", a warning that they would not be able to reread the instructions was added so participants knew this was the final step before starting the study.


Instructions

Please read the following instructions carefully. **You will not be able to return to this page once you begin the experiment!**

Task:

On the next page, you will use an interactive tool to create a personalized college ranking. First, you will use the Build Tool to enter your preferences about colleges in the dataset. Choose as many colleges as you wish and place each one in a category. After you make your selections, the system will provide you with a ranking of the entire dataset of colleges based on your initial choices.

You may return to the Build page to refine your preferences as many times as you like, in order to create a high quality ranking which reflects your preferences.

Create and Revise Your Ranking:

Build: Colleges
Search by names...
Sort
Shuffle
Iowa State University
University of Minnesota--Twin Cities
Ohio State University--Columbus
Georgia State University
University of Colorado--Denver
University of North Carolina--Chapel Hill
University of Maine
Colorado State University
Georgetown University
Clarkson
Oregon State
Michigan State
HIGH
Rochester Institute of Technology
MEDIUM

fig 10. Instructions Module

Originally the Instructions Module only included text, but short animated GIFs demonstrating important features were later added. This improved the Instruction Modules' visual aesthetic by breaking up large chunks of text, and its intuitiveness because a participant would still know how to interact with the tool if they chose not to read the instructions.

Rank Building Module

Participants create partial rankings with either one of three comparison methods: List, Categorical, and Pairwise. For a more in depth description of the comparison methods, refer to *Chapter 2.2.2*. The design of this module was similar to original design of RanKit's Build Tool. The module's display is divided in half, with the data selection pool on the left and the comparison method on the right. Dividing the display like this proved to make the task of ranking much easier for participants.

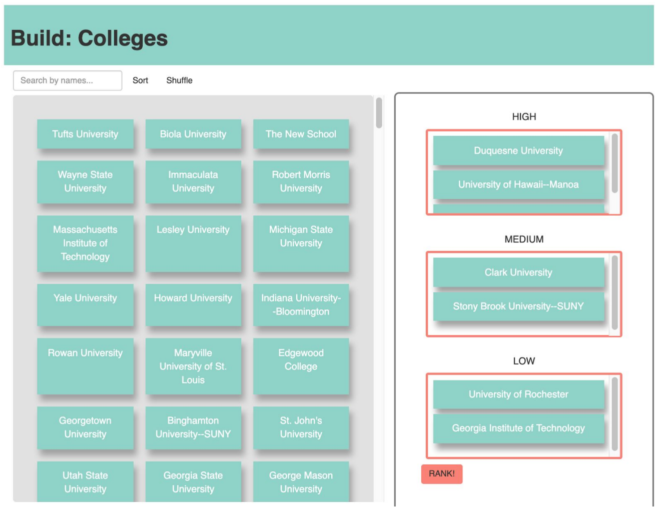
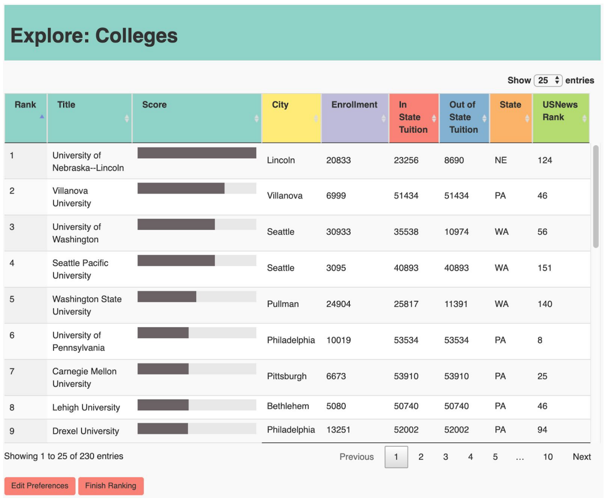


fig 11. Build Module with Categorical Comparison

Furthermore, initial feedback showed that participants were biased to pick the first items that they viewed on the within the data pool, which would give an unfair bias to the colleges that start with "A" [3]. To mitigate this bias, items are randomly shuffled when the data pool is generated. Participants also have the option to sort the data alphabetically, search for specific schools, and reshuffle the data. Participants could get more information on a college by hovering over it, doing so would trigger a popover displaying the attributes the ranking algorithm uses to determine a participant's preferences.

Rank Exploration Module

Participants can view and interact with their final ranking in this module. The data is presented as a table, where participants are able to reorder the final ranking by this and any of given metric. Participants must progress through the entirety of the study to get here, but unlike any other module, it offers the ability to return to to previous module. The ability to return to past modules is not in the original version of Experimentr, and was created specifically for this study. The addition to Experimentr was necessary because we wanted participants to be able to create a partial ranking, and have the ability to edit it after viewing the global ranking. In order to rerank participants must click "Edit Preferences", causing the Rank Building module to reload with all previously chosen items in their correct positions. The button follows the same color scheme as the "Rank" button so participants understand its part of the user study and not the client.


Explore: Colleges

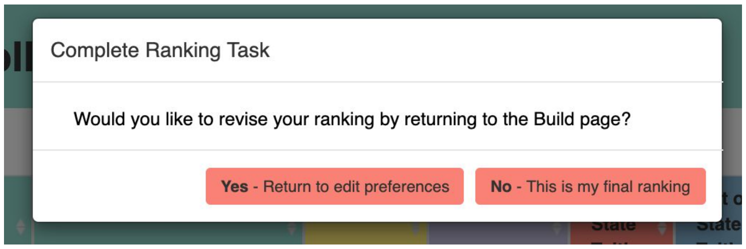
Show 25 entries

Rank	Title	Score	City	Enrollment	In State Tuition	Out of State Tuition	State	USNews Rank
1	University of Nebraska--Lincoln		Lincoln	20833	23256	8690	NE	124
2	Villanova University		Villanova	6999	51434	51434	PA	46
3	University of Washington		Seattle	30933	35538	10974	WA	56
4	Seattle Pacific University		Seattle	3095	40893	40893	WA	151
5	Washington State University		Pullman	24904	25817	11391	WA	140
6	University of Pennsylvania		Philadelphia	10019	53534	53534	PA	8
7	Carnegie Mellon University		Pittsburgh	6673	53910	53910	PA	25
8	Lehigh University		Bethlehem	5080	50740	50740	PA	46
9	Drexel University		Philadelphia	13251	52002	52002	PA	94

Showing 1 to 25 of 230 entries

fig 12. Explore Module

When the participant wants to progress to the final sections of the study, they must click the button “Finish Ranking”. Clicking it causes a small popup window to appear, asking the participant to either confirm their choice to finish ranking or if they would like to edit their partial rankings. Choosing to edit their partial rankings will take the participant directly to the Build module.


Complete Ranking Task

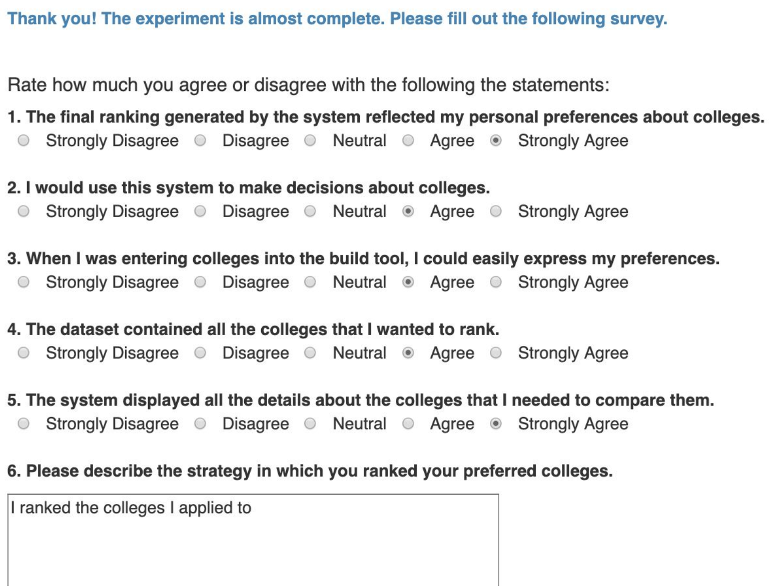
Would you like to revise your ranking by returning to the Build page?

Yes - Return to edit preferences
No - This is my final ranking

fig 13. Popup Confirmation Window

Strategy Module

Although we were interested in general feedback, we wanted quantitative data that reflected the participants experience with the tool. To address this, we tried a variety of different survey scales, but eventually settled on framing questions with the Likert Scale. A survey scale is part of a closed-ended line of questioning that covers a range of opinions on the topic at hand. Likert Scales have a set of either five to seven answers, each of them ranging from one extreme emotion to another, i.e “extremely dissatisfied” to “extremely satisfied”. Questions asked in a Likert-type manner, unlike binary “yes” or “no” questions, allows participants to better express their opinions. Which in turn allows us to better understand the provided feedback, and determine areas of our study to improve.


Thank you! The experiment is almost complete. Please fill out the following survey.

Rate how much you agree or disagree with the following the statements:

1. The final ranking generated by the system reflected my personal preferences about colleges.
Strongly Disagree Disagree Neutral Agree [✓ Strongly Agree]

2. I would use this system to make decisions about colleges.
Strongly Disagree Disagree Neutral [✓ Agree] Strongly Agree

3. When I was entering colleges into the build tool, I could easily express my preferences.
Strongly Disagree Disagree Neutral [✓ Agree] Strongly Agree

4. The dataset contained all the colleges that I wanted to rank.
Strongly Disagree Disagree Neutral [✓ Agree] Strongly Agree

5. The system displayed all the details about the colleges that I needed to compare them.
Strongly Disagree Disagree Neutral Agree [✓ Strongly Agree]

6. Please describe the strategy in which you ranked your preferred colleges.

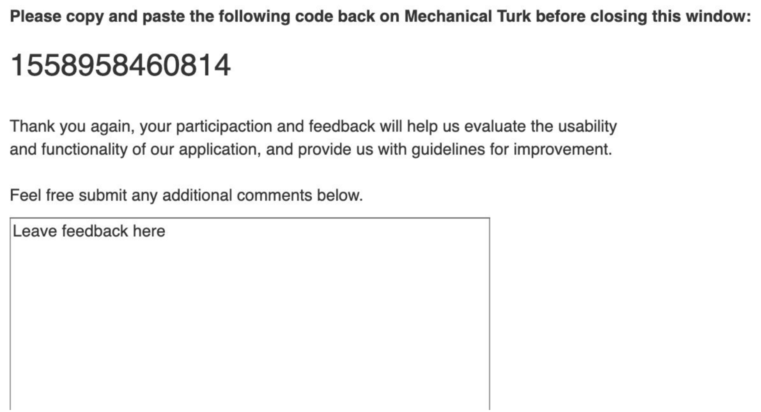
I ranked the colleges I applied to

fig 14. Strategy Module

Additionally, there was a final open-ended question that asked participants to describe their process of ranking colleges. Although this could be inferred from the data collected on their interactions, having a participant's response gives us better, more explicit insight into their thought process. Participants must answer all questions in order to progress to the final module.

Debrief Module

This is the final, and sparsest, module of the study. A code is provided for participants to copy and paste into Amazon Mechanical Turk to receive payment for completing the study. A section is also left for participants to leave general feedback, i.e bugs they encountered. Although encouraged, sharing additional feedback is optional and the study can be completed without providing anything.



Please copy and paste the following code back on Mechanical Turk before closing this window:

1558958460814

Thank you again, your participaction and feedback will help us evaluate the usability and functionality of our application, and provide us with guidelines for improvement.

Feel free submit any additional comments below.

Leave feedback here

fig 15. Debrief Module

3.3.4 Data Logging

One of the biggest factors in choosing Experimentr as the framework for our user study is the ability to log user interactions made within the client. Modifications were made to the way Experimentr originally logged data so that any one of the participant's desired interactions could be logged, and subsequently stored as an entry to a JSON file. Each entry contained a unique ID so data could be easily counted and sorted, this meant we could collect quantitative data on each participants behaviour. We also logged how long each participant spent in each stage of the user study, which comparison model they were assigned, and the answers to their questions. Some of the interactions logged are:

- **Additions** - How many items participants dragged from the data pool into their partial rankings.
- **Removals** - How many items participants removed from their partial rankings, dragging it back into the data pool.
- **Pool Manipulations** - How many times the participant clicks "Sort" or "Shuffle"

• **Partial Rankings** - The participants final partial rankings; every pair and in what order for Pairwise comparison, the list for List comparison, and the each grouping for Categorical comparison .

• **Ranks** - How many times the participant clicked the "Rank!" button, progressing from Rank Building to Rank Exploring.

• **Edits** - How many times the participant the "Edit Preferences" button, returning from Rank Exploring to Rank building.

3.4 Project Development Structure

The team for this project included one undergraduate student: Paul-Henry Schoenhagen, working closely with two mentors: Professor Elke Rundensteiner and graduate student Caitlin Kuhlman. Professor Lane Harrison came on to help advise the team later into the development cycle. Throughout the project Caitlin primarily focused on the backend and the machine learning script, and Paul-Henry was focused on the study interface, the frontend, and integration with Experimentr. The full research team met once a week for an hour. During these meetings, the team presented work completed, received criticism and suggestions, and identified tasks to be completed in the following week.

The graduate student and I would meet weekly to discuss the work we accomplished in the past week and whether we completed the assigned tasks. These meetings often turned into longer work sessions where we would co-develop for multiple hours. This allowed us to perform project merges and effectively work on sections that required help from one another.

In order to better coordinate coding efforts between us, the graduate student shared a pre-existing GitHub repository for the base project of RanKit. A second repository was later created to house the build created for the user study: Rankit_Experimentr. GitHub allowed us to work on different aspects of the same project without impeding each other's progress, this made making and tracking changes more effective. This structure allowed us to collaborate, organize and assign tasks, and stay up to date on the other parts of the project.

Chapter 4: User Study Design

4.1 Goals

Since RANKIT had previously been tested web application, evaluated on its usability and functionality as an application, the purpose of our user study differed. Instead, the focus of our research became how participants behaviour was impacted or influenced by the choice of comparison method. With that in mind we set out to answer the following questions:

- Do participants interact differently with each unique comparison method?
- Are participants satisfied with their global ranking affected by the comparison method?
- Which comparison method motivates users to enter the most amount of data?

Despite each ranking method essentially working the same as Pairwise comparison, having different interfaces to create partial rankings can be more intuitive for the average participant. As in, a participant may have a better understanding of how to rank items in List comparison as opposed to Pairwise comparison, which in turn would affect how they would interact with the tool. We also believe if a user feels more comfortable with the tool, they'll be willing to input more data. In order to substantiate this claim, it's necessary to review and compare the data collected on each participant.

User satisfaction is important to the success of any tool, and to evaluate this goal we will be relying on participant's self-reflection on the experience of the user study. While the tool might be intuitive and easy to use, if participants are unhappy with the final product given, they may not reuse or recommend our tool.

Users tend to want to do as little work as possible. Which can be problematic in this case, since the ranking algorithm requires a user's partial ranking to create the final global ranking. Thus we must find out if one comparison method better motivated participants to enter data than the others, and whether or not it affected their satisfaction. This is imperative in understanding whether or not interactive ranking is effective and accurate with minimal user input.

4.2 Overview

The target audience were people who have a general knowledge of American colleges. Previous experience with ranking and ranking tools is unnecessary. The user study was geared to be easily accessible to an audience that had little to no idea about the task beforehand. The user study focused on individuals in the age range of 20 to 50. People in different professional fields and age categories were targeted:

- Undergraduate students of different majors
- Faculty Members
- Vetted User Study Participants

Subjects either participated in an in-person session conducted by myself, or the online user study hosted by Amazon's Mechanical Turk. In both cases, participants iterated through Rankit_Experimentr as intended.

4.3 Pilot User Study

There were three participants in this pilot user study: one undergraduate student and two faculty members. The goal of this user study was to test the stability of Rankit_Experimentr before deploying the final build online, refine both our studies instructions and post-test questions, and gauge the user's understanding of the tool before moving on.

The sessions were conducted with one interviewer and one participant. Each participant was given a machine with the current build of Rankit_Experimentr running and a brief description of the task at hand. After reading and agreeing to the consent form, participants were interact with the tool with minimal input or feedback from the interviewer.

Participants were each assigned a different comparison method, and the sole task was to used the assigned comparison method create a ranking using a limited amount of data points. Throughout the process the interviewer took notes about the participant's experience interacting with the tool, and the participant was encouraged to talk aloud about what they were doing.

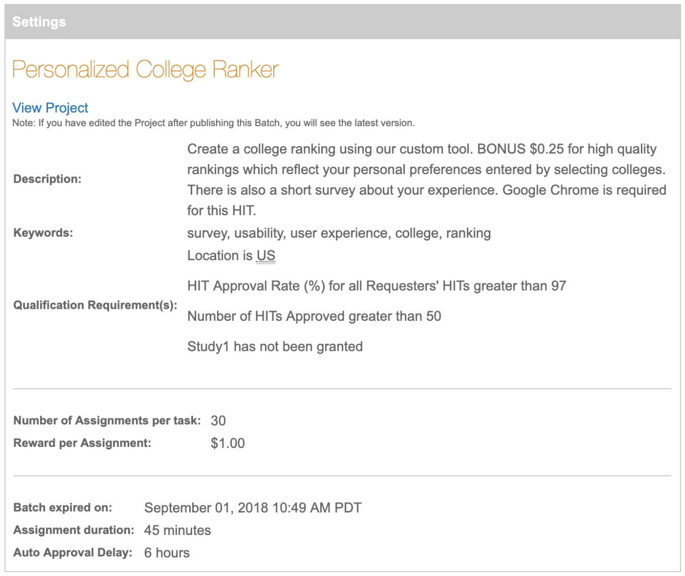
When the participant was finished iterating through Rankit_Experiment, the interviewer then asked the post-test questions and recorded answers. The study was deemed finished when the participant answered the final question. Responses from this group were expected to be qualitative and focused on the features and functionality of Rankit_Experimentr.

4.4 Online User Study

This project was inherently built to be an online user study completed remotely by participants online. Much of the projects pacing, flow, and functionality was built around the idea that participants would interact and complete the user study on their own accord without the need for an interviewer.

Amazon Mechanical Turk (MTurk) is an online crowdsourcing platform where both individuals and companies can have an online workforce perform desired tasks virtually. Although it may not have been its original intention, MTurk is a valuable platform for user studies specifying in human-computer interaction. Mturk is split into two types of users: Requesters and Workers. For the purpose of our experiment we created a Requester account, which allowed us to place our user study on MTurks public marketplace for Workers to complete. Although our study was available via MTurk, it was hosted on WPI's servers.

Mturk provides a reliable way to gather a substantial amount of participants to complete the user study. We were able to guarantee each Worker was experienced since each Worker has qualifications associated with their account, such as HIT approval rate, and we were able to make our study inaccessible to inexperienced Workers. Steps were also taken to ensure Workers could not participate in the user study multiple times.


Settings

Personalized College Ranker

View Project

Note: If you have edited the Project after publishing this Batch, you will see the latest version.

Description:	Create a college ranking using our custom tool. BONUS $0.25 for high quality rankings which reflect your personal preferences entered by selecting colleges. There is also a short survey about your experience. Google Chrome is required for this HIT.
Keywords:	survey, usability, user experience, college, ranking
Qualification Requirement(s):	Location is US HIT Approval Rate (%) for all Requesters' HITs greater than 97 Number of HITs Approved greater than 50 Study1 has not been granted
Number of Assignments per task:	30
Reward per Assignment:	$1.00
Batch expired on:	September 01, 2018 10:49 AM PDT
Assignment duration:	45 minutes
Auto Approval Delay:	6 hours

Fig 16. User Study Mechanical Turk Settings

Worker compensation is required for MTurk, and after initial tests showing the average time of completion being between 5 - 8 minutes, we structured payment as a base payment of $1.00 and a $0.25 incentive for creating a satisfactory ranking. Unknown to the Workers was that they each received the $0.25 incentive upon completion.

In order to make substantiated claims about our ranking system we wanted recurring results over a large group of people. Amazon Mechanical Turk serves the purpose of getting many vetted individuals to complete our user study in a reasonable timeframe. Due to its reputation, ease, and integration with Experimentr, MTurk was decided to be the best service to use for carrying out our online user studies.

4.4.1 Deployment Practices

Occasionally during deployment a participant would encounter a bug which forced them to exit the user study before finishing. This was an issue since Workers needed the code on the final module for payment. This was especially troublesome since the measures we had to prevent duplicate workers, also meant they could not retake the study in the case of a session-ending bug. In these cases Workers would reach out via email or MTurk, a hotfix was applied, and a study specifically for that Worker was put on MTurk.

Chapter 5: Evaluation

5.1 Data Evaluation Process

One drawback of logging every user interaction is the sheer amount of data collected on each participant. First the data collected is pulled from the Redis server and stored in a JSON file, the file contains all fields collected on the frontend, fields such as Worker ID, time elapsed, user interactions etc. Experimentr does not enforce a data schema, so entries within the JSON file may have missing values. Afterwards we implemented a python script that cleaned the data, sorting it by Worker ID and placing desired data into corresponding columns, and converted it from a JSON to a CSV file. CSV files are similar to spreadsheets, and while they may not be the best for long-term data storage, their structure is organized in a simple manner making it efficient to run evaluation on. The cleaning script also separates out and formats the open ended questions so they're more readable. There were then three separate scripts to evaluate:

- Time - time elapsed, hourly rates
- Interactions - Additions, Removals, Pool Manipulations, Partial Rankings, Ranks, and Edits
- Questions - Participants responses to the Likert Scale.

5.2 Onsite User Study Results

This section outlines the results from the pilot onsite user study, although small in size this user study provided important initial insight into how participants behave and react to the study. Rankit_Experimentr was actively logging user interactions during the onsite interviews, but due to issues during the pilot run, the data was deemed unreliable.

Topics that arose from the pilot study, either from participant feedback or tool malfunctions, and how they were addressed before deploying the final study online are:

- **Weight Tracking Removed** - amongst the data originally collected on the user were the weights the ranking algorithm calculated for each attribute. Although helpful in theory, it unnecessarily bloated log records and significantly slowed application performance.
- **Datapool Scrollbar** - two of three participants requested a scroll bar within the datapool so they could keep track of their position within the pool. Although initially included, this functionality was lost while merging Rankit and Experimentr, but was ultimately re added.
- **Instructions Refined** - After hearing the participants read the instructions aloud and fielding subsequent questions, refinements were made to sections participants had trouble understanding and animated GIFs were updated.
- **Post-Test Questions Refined** - Questions were changed to be posed in the first person, adherent to the Likert scale, and were refocused to gain better insight into the participants thought process. This was especially important for close-ended questions where users wouldn't be able to fully express themselves.

5.3 Online User Study Results

Over time we amassed 144 participants. The study randomly assigned 49 participants to rank using the List Comparison method, 45 to use the Categorical Comparison method, and 50 to use the Pairwise Comparison method. Images in this section also appear in a paper written by PhD Caitlin Kuhlman [11], with whom I have collaborated throughout this MQP project. I am grateful for her allowing me to utilize the below images.

5.3.1 User Interactions

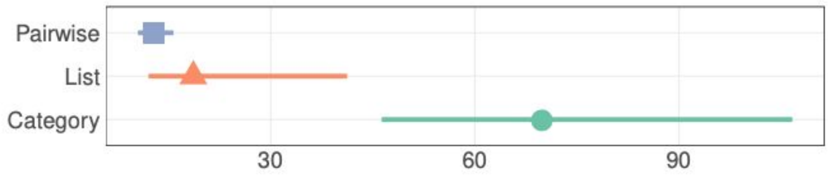


fig 17. Items Added

Here it's demonstrated that when assigned categorical comparison, participants interacted with more items from the dataset. The average amount of items added during categorical comparison was 69.9, an average of 18.6 items were added during list comparison, and an average of 12.7 items were added in pairwise comparison. As shown in figure 17**,** we can interpret that the average participant entered at least 5 and at most 95 additional items in categorical comparison than the other two methods.

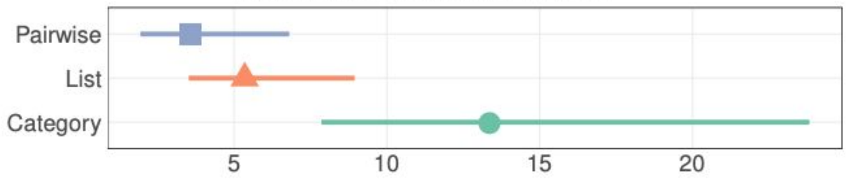


fig 18. Items Removed

In addition to keeping track on the amount if items added, we kept track of the amount of items removed from a users partial ranking, as shown in *figure 18*. The average amount of items removed during categorical comparison is 13.4 items, an average of 5.3 during list comparison, and an average of 3.6 during pairwise comparison. Although there were less removals on average, the trends previously observed with additions are still apparent.

Although it's easy to assume from this data that participants using categorical comparison entered in significantly more amount data than the other two methods, one must consider inherent interactivity of each method. With list comparison the participant creates a descending-order sublist, in pairwise they create pairs of objects, with categorical there are three separate bins to for them to place an item in. And with the current datalogging, moving an item from the one bin to another bin would register a removal from the first bin and and addition to the second bin.

5.3.1 Time Spent Interacting

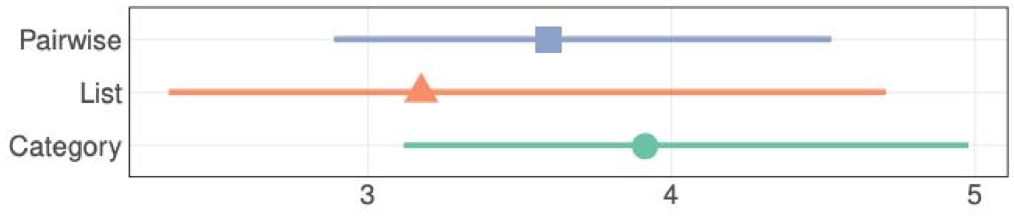


Fig 19. Time Spent Building (minutes)

Even though categorical comparison led by a wide margin in items added and removed, we do not see this same margin repeated in time spent building. Participants assigned categorical comparison spent an average time of 3.9 minutes building, those assigned list comparison spent 3.6 minutes, and those assigned pairwise spent 3.2 minutes. Thus meaning the comparison model does not have a significant effect on the time spent building.

Even though the building module's main task is to create partial rankings from the dataset, there is an inherent second task: searching for and selecting colleges to rank. As it stands there is no way to determine what percentage of time spent on the build module was spent searching for items to rank versus ranking those same items. Although it may seem insignificant, knowing how long a participant interacted with our ranking interface would provide better insight.

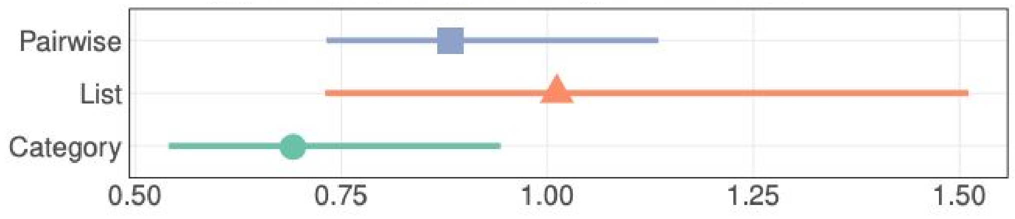


Fig 20. Time Spent Exploring (minutes)

Participants actually spent the least amount of time exploring the final ranking when assigned categorical comparison, but overall time spent exploring was not significantly affected by which comparison method was assigned. Those assigned list comparison spent an average time of 1 minute build, those assigned pairwise spent 0 .9 minutes, and those assigned categorical spent 0.7 minutes. It's possible people spent more time exploring with the list comparison method due to it being most similar in structure to how the global ranking is presented.

5.3.1 User Satisfaction

The results to the Likert-style statements are provided in figure 21. The three statements are as follows:

A. The final ranking generated by the system reflected my personal preferences about colleges

B. I would use this system to make decisions about colleges

C. While I was entering colleges into the build tool I could easily express my preferences

Overall sentiments ranged from neutral to mostly positive, and again we see the comparison method does not have a significant impact on participants reported satisfaction. Which is a little odd considering the difference in amount of interactions.

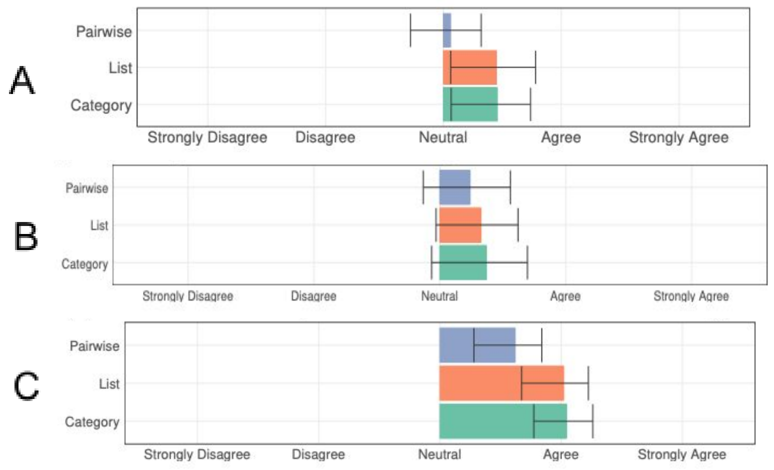


Fig 21. Responses to Qualitative Statements

Participant responses were read and evaluated in an effort gain more understanding, a sample of responses can be viewed in *Appendix B*. As is with quantitative data, when evaluating qualitative data it is important to determine what is and isn't useful. This isn’t to suggest responses that were unfavorable were disregarded; genuine grievances and constructive criticism is needed to improve our tool. Unfortunately due to the relative anonymity of the internet, some responses were clearly made in jest and provided no real value. After filtering and evaluating the responses we were able to confirm a positive response from users, despite comparison method, as well what information users find important to ranking. Further discussion on this topic can be viewed in *section 6.2.2*. A common strategy for users was choosing colleges they were fond of, as opposed to picking schools that were nationally renown, making it easier for them to determine if their final ranking reflected their submitted preferences.

Chapter 6: Conclusion

6.1 Summary

In the age of the Internet, there is an exceptional amount of data being constantly uploaded, shared and viewed. An overwhelming amount of data makes it a difficult task for a single person to find threads of similarity in all of it. Making the ability to find and rank data, based on its relevance to you, exceedingly cumbersome. Furthermore rankings released by institutions and media outlets are not always useful to an individual, partly because they are designed for the masses instead of the individual, and partly due to the complexity of ranking a large dataset. While there were plenty of challenges uncovered during project development, we chose to tackle them due to the far-reaching implications on the future of mixed-initiative ranking systems.

Over the course of this project, our team successfully completed the key tasks we initially set out to complete. We have:

- Successfully integrated a pre-existing project; RANKIT, with web framework Experimentr.js. Thus creating Rankit_Experimentr: a modern web application with a Python and Redis backend. It's focal point being the Rank and Explore modules, which users are able to create and edit their personal rankings. With one the main features of the tool being the data collection which allows for efficient recording and studying of user behaviour.

- Conducted multiple user studies with the intent to study how users interacts with our tool.

Although there is much room for our tool to improve, we have created a worthwhile testing environment for future iterations of RANKIT, and evaluated the merits of each comparison model. Ways to further the advancement of our tool are outlined in the following section.

6.2 Future Work

6.2.1 Machine Learning Script

Improvements to the machine learning script were made throughout the progress of the project, but latency still proved to be an issue an impeded progress on a few occasions. Further research into improving the amount of partial data it can process, and overall performance of the script.

6.2.2 Dataset Refinement

Participants' responses to our open-ended question provided us with insight into how participants ranked items, information they find important and ways to improve the dataset. Though users did care about quantitative attributes, such as class size and tuition rates. Many of them reported ranking based off qualitative features such as: sports, weather, type of school (liberal arts vs engineering) , perceived political alignment etc. Adding some qualitative attributes would provide a user with more information, better way to express their preferences, and more points of similarity to determine if the final ranking is satisfactory.

6.2.3 Comparison Method

We currently have data showing how our comparison methods stack up to one another, but how does it compare to pre-existing systems? Future work should also include the creation and inclusion of a comparison method similar to pre-existing systems, such as LineUp and Podium. This would allow for a closer comparison between ours and competing ranking systems, and insight into which provides better data elicitation and user satisfaction.

6.2.4 Motivators

Users tend to want to put in the least amount of effort possible, which proves to be an issue with semi-supervised ranking problems which depend on user input. Ideally our system would elicit data from users without it feeling like a chore. Technical constraints prevented us from implementing real-time motivators in Rankit_Experimentr that were included in RANKIT. Future work could be re-implementing the motivators, and conducting another user study.

Evaluating the results would reveal if the inclusion of motivators creates a meaningful impact on both user satisfaction and input, thereby proving if it is an effective form of data elicitation, and a feature that users find useful.

References

[1] About Node.js. (n.d.). Retrieved from https://nodejs.org/en/

[2] Datasets | Kaggle. (n.d.). Retrieved from https://www.kaggle.com/datasets

[3] Deva, G., Doherty, D., Nurbekova, M., …. Phyo, Z.(2018). RANKIT: Designing Interactive Tools for Personalized Ranking Analysis.

[4] E. Wall, S. Das, R. Chawla, B. Kalidindi, E. T. Brown, and A. Endert. Podium: Ranking data using mixed-initiative visual analytics. IEEE transactions on visualization and computer graphics, 24(1):288–297, 2018.

[5] Fabian Wauthier, Michael Jordan, and Nebojsa Jojic. 2013. Efficient ranking from pairwise comparisons. In International Conference on Machine Learning. 109-117.

[6] Feng, M., Deng, C., Peck, E. M., & Harrison, L. (2018). The Effects of Adding Search Functionality to Interactive Visualizations on the Web. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI 18*. doi:10.1145/3173574.3173711

[7] Gratzl, S., Lex, A., Gehlenborg, N., Pfister, H., Streit, M. (2013). LineUp: Visual Analysis of Multi-Attribute Rankings. IEEE transactions on visualization and computer graphics, 19(12).

[8] Introduction to Redis. (n.d.). Retrieved from https://redis.io/topics/introduction

[9] IRB-FAQs. Retrieved from https://www.fda.gov/regulatory-information/search-fda-guidance-documents/institutional-review-boards-frequently-asked-questions#IRBOrg

[10] Krill, P. (2015). A Developer's Guide to the Pros and Cons of Python. Retrieved from https://www.infoworld.com/article/2887974/application-development/a-developer-s-guide-to-the-pro-s-and-con-s-of-python.html

[11] Kuhlman, C., Doherty, D., Nurbekova, M., Deva, G., Phyo, Z., Schoenhagen, P., . . . Harrison, L. (2019). Evaluating Preference Collection Methods for Interactive Ranking Analytics. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI 19*. doi:10.1145/3290605.3300742

[12] Kuhlman, C., Rundensteiner, E., Neamtu, R., Ahsan, R., Stokes, J., Hoxha, A., ... & Rangan, R. (2017). Towards an Interactive Learn-to-Rank System for Economic Competitiveness Understanding.

[13] Lane Harrison (2018). experimentr: a hosting/data-collection backend and module-based frontend for web-based visualization studies. V1.0.0, https://github.com/codementum/experimentr. doi: 10.5281/zenodo.2564082

[14] Learning to rank - RLScore 0.7 documentation. (2016). Retrieved from staff.cs.utu.fi/~aatapa/software/RLScore/tutorial_ranking.html#learning-to-rank

[15] Martin Szummer and Emine Yilmaz. 2011. Semi-supervised learning to rank with preference regularization. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management. ACM, 269-278.

[16] MATTERS. (n.d.). Retrieved from http://matters.mhtc.org/

[17] Movies Dataset. (n.d.). Retrieved from https://www.kaggle.com/rounakbanik/the-movies-dataset

[18] National University Rankings. (n.d.). Retrieved from http://www.usnews.com/best-colleges/rankings/national-universities

[19] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 133-142.

Appendix A: Consent Form

Informed Consent Agreement for Participation in a Research Study

Investigators: Paul-Henry Schoenhagen
Contact Information: Email at pmschoenhagen@wpi.edu
Title of Research Study: RanKit: Designing Interactive Tools for Ranking Analysis

Introduction: You are invited to participate in a research study to evaluate the usability and functionality of a ranking application. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

Purpose of the study: Many existing rankings target general audiences, yet cannot take into account individual preferences. We have developed a web application to help users customize and interpret rankings to meet their own end goals. The purpose of this study is to evaluate the usability and functionality of our application and to get feedback for improvement.

Procedures to be followed: Your participation will consist of a single Mechanical Turk HIT where you will be asked to create a ranking using our ranking application. Afterwards, you will be asked to rate the overall quality of the interaction using our questionnaire. There are no right or wrong answers. The questionnaire should take approximately 5 - 10 minutes to complete.

Risks to study participants: This study involves the following risks: minimal fatigue and eyestrain from looking at a screen. There may be other risks that we cannot predict. In case of discomfort, you may choose to discontinue your participation at any time without penalty.

Benefits to research participants and others: You will be paid for each HIT completed, but other than that: there are no direct benefits for you to participate in this study.

Record keeping and confidentiality: Records of your participation in this study will be held confidential so far as permitted by law. However, the study investigators, the sponsor or it's designee and, under certain circumstances, the Worcester Polytechnic Institute Institutional Review Board (WPI IRB) will be able to inspect and have access to your Amazon account information during data collection. This information will be stripped from the final dataset. Your

individual answers will not be published; end results will be aggregated before publication. Any publication or presentation of the data will not identify you.
Compensation or treatment in the event of injury: No compensation will be made for injuries resulting from participation in this study. You do not give up any of your legal rights by signing this statement.

For more information about this research or about the rights of research participants, or in case of research-related injury, contact: You can contact the team in writing an email at pmschoenhagen@wpi.edu or the Principal Investigator, Elke Rundensteiner, at rundenst@wpi.edu. In addition, you may also contact the chair of the WPI Institutional Review Board (Prof. Kent Rissmiller, Tel. 508-831-5019, Email: kjr@wpi.edu) or WPI's Human Protection Administrator (Gabriel Johnson, Tel. 508-831-4989, Email: gjohnson@wpi.edu).

Your participation in this research is voluntary. Your refusal to participate will result only in loss of payment from Mechanical Turk. You may decide to stop participating in the research at any time.

By clicking "I Agree", you acknowledge that you have been informed about and consent to be a participant in the study described above. You are entitled to retain a copy of this consent agreement.

Appendix B: Sample Participant Responses

- I put in all the colleges I knew and how I thought they compared.
- I was looking for top schools, plain and simple
- I went by name, state, city. I know all the top schools so I placed them in the high category. The state universities also went into the high category. The city universities went into medium and the small local ones went into low.
- I ranked colleges I didn't know the location of into medium. I ranked colleges in liberal areas low. I ranked conservative area colleges high.
- I ranked based on prestige colleges
- I ranked the colleges I knew I would like to physically attend based on the look of the campus and where in the US it was located. I selected the universities I knew a lot about and knew I would be interested in attending.
- I looked at ones I was familiar with, plus some places I would like to live. Then I looked at ranking plus tuition.
- I already had a good idea of what I want in a college before I started it. I went to a small, private college and stayed away from bigger, massive universities. Most, if not all, of these colleges are very big. So I couldn't easily rank them by that (my college you could walk across campus in 10 minutes, slow walking!). That said, I tended to rank the best ones in cooler weather climates (I hate warm/hot weather). So you won't see anything in Florida on my list. I'm also very liberal, so I try to avoid conservative places (though I did rank some higher that are in conservative states due to their liberal town setting). I also went more for some of the colleges that are lesser known, but still really good (i.e.- Clark University). I would fit in better places like that. So that was my general strategy! Fun HIT!

- In the colleges are need to compare them self to the other college. all of the colleges wanted to the rank. The ranking system is generated by the reflection of the personal preference about colleges.
- I used the US rankings
- I saw the US rank and I know some of the top rated colleges. These are my strategies for selecting the colleges
- I was looking for a particular major.
- I picked the two I knew something about
- Personal experience or based on what information I knew about them.
- What I know about the colleges.
- i chose colleges in states i love as my highest priority. second were colleges i have either heard of or colleges that are well known and ranked previously. in the low category i put colleges that are in states i would never want to go to.
- I went by the colleges i actually knew something about. i live in LA so i know tulane and LSU are good. i grew up in upstate NY so i know albany and binghamton are decent. and i put a few i had never heard of in the last category.
- I wanted to compare colleges that I thought were similar, and that have interactions with each other. I also wanted to make one of the sides clearly larger to not skew results. I wanted to really get a sense of the full spectrum. I felt in this way it would broaden the rankings for me.
- academics, prestige, and sports
- I ranked colleges that I was personally familiar with. I think there needs to be more data for each college like graduation rate and tuition rates.
- I tried to focus on colleges near my area (Chicago) and more famous universities like Harvard. These are the institutions I know the most about.

- Close to home, or ones in areas I would like to go. Most of the colleges I actually wanted to go to weren't listed.
- I ranked the colleges I knew I would like to physically attend based on the look of the campus and where in the US it was located. I selected the universities I knew a lot about and knew I would be interested in attending.
- I went by name, state, city.
- this colleges accurate rank conversation
- I went with schools that were not overly expensive while also being highly regarded within the state. State schools would generally be on the lower end as they were generally expensive and not the best for students
- I ranked based on what I have heard about different schools, seen on the news and athletics wise. Also from my most to least favorite.
- I ranked them High, Medium, and Low based on the ranking they were given by the third party ranking.
- 1-50 was high. 51-100 was medium. 100+ was all low.

Appendix C: Rankit_Experimentr Manuscript

Intro Blurb

College rankings are a valuable tool used by many students when they are considering whether to apply to a certain school. However, published rankings do not take into account the preferences of individual students.

Task:

Your task is to create a personalized college ranking. On the following pages, you will be presented with a dataset of colleges, and asked to input your preferences about some of them. From this partial information that you provide, the system will learn a global ranking over all the items in the dataset. Then you will be asked to give feedback on your experience and understanding of the ranking process.

Training Script

Please read the following instructions carefully. **You will not be able to return to this page once you begin the experiment!**

Task (Categorical Comparison):

On the next page, you will use an interactive tool to create a personalized college ranking. First, you will use the Build Tool to enter your preferences about colleges in the dataset. Choose as many colleges as you wish and place each one in a category. After you make your selections, the system will provide you with a ranking of the entire dataset of colleges based on your initial choices. You may return to the Build page to refine your preferences as many times as you like, in order to create a high quality ranking which reflects your preferences.

Create and Revise Your Ranking (Categorical Comparison):

Drag colleges from the dataset to the left to place them in a category on the right. Place your top choices in the “High” category, your least preferred colleges in the “Low” category, and colleges in the middle in the “Medium” category. For each pair, place the preferred college on the left under “High” and the less preferred college on the right under “Low”. Each college can be used in multiple pairs. Click "Rank" to see your generated ranking. Click "Edit Preferences" to go back to the Build tool and refine your ranking further.

Task (List Comparison):

On the next page, you will use an interactive tool to create a personalized college ranking. First, you will use the Build Tool to enter your preferences about colleges in the dataset. Choose as many colleges as you wish and place each one into a ranked list. After you make your selections, the system will provide you with a ranking of the entire dataset of colleges based on your initial choices.

You may return to the Build page to refine your preferences as many times as you like, in order to create a high quality ranking which reflects your preferences.

Create and Revise Your Ranking (List Comparison):

Drag colleges from the dataset on the left to the list on the right. Place the most preferred colleges at the top of the list, and the rest in descending order of your preference. Click "Rank" to see your generated ranking. Click "Edit Preferences" to go back to the Build tool and refine your ranking further.

Task (Pairwise Comparison):

On the next page, you will use an interactive tool to create a personalized college ranking. First, you will use the Build Tool to enter your preferences about colleges in the dataset. Choose as many colleges as you wish and arrange them as pairs. After you make your selections, the system will provide you with a ranking of the entire dataset of colleges based on your initial choices.

You may return to the Build page to refine your preferences as many times as you like, in order to create a high quality ranking which reflects your preferences.

Create and Revise Your Ranking (Pairwise Comparison):

Drag colleges from the dataset on the left to form pairs on the right. For each pair, place the preferred college on the left under "High" and the less preferred college on the right under "Low". Each college can be used in multiple pairs. Click "Rank" to see your generated ranking. Click "Edit Preferences" to go back to the Build tool and refine your ranking further.

Interact with the data:

- Mouse over an individual college to see its data attributes.
- Click the "sort" button to order the colleges alphabetically.
- Click the "shuffle" button to randomize which colleges you see.
- Use the search bar to search for specific colleges.

Share your ranking strategy:

When you are finished with the ranking tool, you will be asked to describe your findings, and answer a few questions on your experience with this interactive rank building system