Worcester Polytechnic Institute

RANKIT: Designing Interactive Tools for Personalized Ranking Analysis

An Interactive Qualifying Project
Submitted to the Faculty of WORCESTER POLYTECHNIC INSTITUTE
in partial fulfilment of the requirements
for the Degree of Bachelor of Science
3 March 2018

By
Goutham Deva, Diana Doherty
Malika Nurbekova and Zarni Phyo

Report Submitted to:
Professor Elke Rundensteiner

PhD Student Mentor:
Caitlin Kuhlman

Acknowledgements

We would like to thank the following individuals and organizations for their support and assistance throughout our project:

• **Professor Elke Rundensteiner,** our advisor from WPI for giving us the opportunity to work on this project as well as guiding us through every phase of the project.

• **Caitlin Kuhlman, WPI,** for her guidance and support in familiarizing us with resources and tools to help support the development of the ranking tool.

• **MaryAnn VanValkenburg, WPI,** for her assistance in the research component, assets recommendations, and analysis of the ranking tool code as well as providing useful feedback on the project.

• **Collaborative Research Experiences for Undergraduates (CREU),** for assisting us with funding opportunities and research guidance for us to complete this project.

• **Worcester Polytechnic Institute,** for providing us the education and opportunities needed for us to complete this project.

• **CREU,** for funding the research required for creating a user friendly and fair ranking system.

Table of Contents

Table of Figures

Abstract

Rankings are commonly used to evaluate everyday decisions. However, constructing rankings can be difficult because building a ranking requires extensive knowledge about the observed dataset. To overcome this problem, we present RANKIT, 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. Users can then inspect the weights and attributes of their overall ranking. Intuitive interfaces allow for the effective building and exploring of rankings. The system was evaluated with a comprehensive two part user study consisting of online surveys and in-person interviews. Results reveal that RANKIT succeeds at both informing users and sparking their interest about personalized rankings. Future improvements have also been identified to make RANKIT even better.

Executive Summary

Ranking and Existing Tools
Nowadays, rankings released from institutions and media are not always useful to individual users, partly due to over-generalization of interests, as well as the complex nature of ranking data itself. Yet to find relevant data and rank them according to the user's preference has become a crucial part of everyday life.

This project focuses on building a framework that assists users in ranking of data. It evaluates and compares existing ranking tools, such as MATTERS, Lineup, and Podium, and discusses how these technologies have informed your design and development of the RANKIT tool.

The goals of the research are to understand how users rank data, then to design and evaluate an intuitive way of building partial rankings, and to learn about the amount of partial information users are willing to give in establishing a personalized ranking.

Design and Implementation of RANKIT Technology
An intuitive visual design is a vital part of this research. RANKIT went throughout numerous iterations of the development process to address user problems with the original usability of the tool. The team built RANKIT, a web application with a Python backend. Users provide a particular form of partial rankings, and the system then learns the complete ranking using machine learning techniques.

Evaluation using a User Study and Results
The user study was divided into two parts: in-person interviews and an online survey. The in-person interviews were conducted on two study groups, including college students and human-computer interaction experts. The online survey was conducted on people of different social groups.

According to results of the two parts of the study, over 60 percents of the users rated RANKIT intuitive. This was measured based on collected average of intuitiveness of the separate features and web pages.

The in-person portion of the study showed that people used the List Comparison method the most when asked to choose among three methods that they were already familiar with. However, when asked about the intuitiveness of each method, the majority of the in-person participants rated the Categorical Comparison method to be the most intuitive while the online survey

respondents rated List Comparison method to be the most intuitive. Over 80 percents of the users interact with 2 to 8 objects, but were willing to rank up to 16 objects to get better results.

However, we found that when datasets contain hundreds of objects each, 16 partial rankings is not sufficient enough to produce a confident ranking. Therefore, future work needs to be conducted improving usability to increase the number of partial rankings the user is willing to provide. The method that suffers the most in accuracy from small partial ranking numbers is Pairwise Comparison because users need to manually insert all combinations of rankings that Categorical and List Comparisons take care of automatically. The main criticism of RANKIT was its non-intuitive interface on mobile applications. However, the application was designed to be used on a laptop or a desktop, and the mobile interface was never taken into account. Future work can expand into making RANKIT a mobile friendly application.

1. Introduction

Motivation

Modern advancements in processing and storage of information have inspired research into different strategies to consume versatile data and synthesize it into simple models. The evaluation of such models can be found in services that help industries mitigate risk, make smart decisions, and gain better insight into different choices. One such model, ranking, allows individuals to weigh and observe in clearer detail the choices they are faced with, guiding them to optimal decisions.

Rankings are employed in everyday life and decision making. Be it a problem of selecting the ideal college, deciding which restaurant to have dinner at, or what movie to watch, the concept of ranking one item over the other is a common tool for decision making. Sometimes, however, the result of a ranking algorithm on decisions are not as evident. In search engines, for example, a webpage being served in the top results is a matter of being visited or not, and a matter of having influenced one's opinions or play a part in confirmation bias. Banks rely on credit scores to determine whether a loan for a car, a house or college tuition is likely to be paid, thereby rejecting or approving a loan.

Challenges

There are limitations to ranking. For a ranking to be accurate in expectations, it is important to find valuable information in incomplete data. The data analyzed derived from real world events is never perfect. A collection of data might be missing or an outlier may appear to majorly affect a basic observation. Data must be cleaned before it can be synthesized. Despite the preprocessing, the data could still be computationally impossible to analyze in reasonable amount of time, too partial to derive any practical assumptions on, or misleading in secluded context.

Shuffling through the disorganized and incomplete data might be impossible for individuals looking for a useful ranking as it either requires extensive knowledge in the field or the usage of software tools that are not readily accessible. As a resolution, companies simplify the process of ranking and publicize the end result - the ranking. However, when interpreting multi-attribute datasets, a slight change of weight between attributes can heavily impact ranking results. Most publicly available rankings do not expose the attributes used to compose the ranking. This lack of disclosure results in possible exploitation of data because rankings can imply a conclusion that is not necessarily true.

It is important to be wary of rankings without disclosure of weights and their attributes. To deflect this problem, it is vital to encourage individuals to formulate their own rankings. A tool that would allow users to come to their own ranking can be extremely valuable as it resolves the problem of making sense of big data and allows users to rank by their personal preferences, instead of someone else's (Gratzl-Streit et al., 2013; Wall-Endert et al., 2013).

Manually ranking, or assigning each attribute a specific weight, can be a difficult task for the individual who is unfamiliar with the whole dataset or is unsure about the importance of each attribute. Coming up with the weights requires prior knowledge, which is not friendly to beginners. Often rankings are created with no knowledge of the relative importance of attributes. Therefore, it is important to construct rankings that allow users to apply their intuition about the order of certain items within a dataset to then automatically determine which attributes ought to be considered more valuable than others. Sophisticated techniques employed through machine learning have been developed for information retrieval (Liu, 2009) which allow for automatic rankings of datasets. Though there are powerful machine learning techniques used in information retrieval and recommender systems, most users do not have access to them. The need for interactive, highly intuitive tools which give users the ability to leverage learning-to-rank algorithms was the inspiration for RANKIT.

Proposed Solution

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. Intuitive interfaces allow for the effective building and exploring of rankings. The key innovation of RANKIT is the Build tool, which provides three alternative interfaces for users to input their knowledge about items to be ranked.

The Build tool collects this partial ranking information from users based on their domain of knowledge or priorities and leverages Pairwise Comparisons to automatically derive rankings on the backend using a leaning-to-rank algorithm. In a pairwise comparison method, the goal is to minimize the number of inversions in a rank. In other words, minimize the cases where the pair of results are in the wrong order relative to the ground truth (Liu, 2009). This approach allows RANKIT to produce accurate rankings based on users' needs without exhaustive input from the user.

Once the ranking is built, users can pass the result to the Explore tool. The Explore tool, in turn, visualizes the ranking and the underlying attributes used to create it. To encourage transparency in ranking, Explore allows users to view attribute weights and their relative contribution to the overall ranking.

During the design process for RANKIT, unofficial questionnaires and interviews with user interface experts were conducted to ensure that RANKIT satisfies consumer demand. And, to incorporate this feedback into the project, RANKIT went through multiple iterations, each of which strived to achieve improvements that would take RANKIT to the competing grounds of previous state-of-the-art..

While the core computations of RANKIT are derived from pairwise comparisons, the evaluation revealed that other methods were more intuitive. Therefore the Build tool was expanded to have three different options for constructing a ranking. These options are referred to as comparison methods, which are methods that allow ranking of one item over one or more other items.

The first method is List Comparison. In this method, users present their preferences by sorting rankable items in an ordered list. The first item is considered to be ranked as number one, the second item is ranked as number two, the third is three and so on. The second ranking method is Categorical Comparison. In this method users to express their preference by sorting items into three categories: high, medium, low. Items within each categories are not ranked, however, an item in the high category is better than all items in both the medium and the low categories, and an item in the medium category is better than all items in the low category. The last ranking method is Pairwise Comparison, where users directly express their preferences as relations between pairs of items All three methods are reduced to pairwise comparisons under the hood and sent to the machine learning algorithm on the backend.

We hope that RANKIT will stimulate a greater interest in users because it delivers a better understanding of rankings and informs how ranking results are produced in the real world.

2. Background

2.1 Problems in Ranking

Limitations

A ranking is a way to simplify the process of deciding between objects. In any non-trivial circumstance, objects have multiple attributes that add to or diminish their merit, and it is too difficult for an individual to determine which object is best. For example, consider the task of choosing a college to attend. Relevant attributes to consider include the cost to attend, the student-to-teacher ratio, and median salary of students after graduation. It is unlikely that any one college is the cheapest, has the lowest student-to-teacher ratio, and has the best-paid graduates, so making a reasonable choice involves considering the relative usefulness of these attributes. Ranking facilitates this by assigning weights to each attribute that control how much it affects the choice. As a result, a small difference in a heavily-weighted attribute would contribute more to a decision than a lightly-weighted attribute.

Ranking makes intuitive sense; the benefits of some attribute outweigh the costs of others. Ranking, however, also has limitations. For one, it requires attributes to be encoded in a way that can be plugged into a ranking function. For another, the resulting rank of the objects does not provide information as to how *much* better one object is than another.

One limitation to ranking is that it has the potential to oversimplify attributes. Consider, for example, the cost to attend. One could encode this attribute as a continuous variable, such as the number of dollars to attend, or discretize it into categories, such as less than $40,000 and greater than $40,000. The continuous encoding is problematic because quality of education might not be linearly related to cost; is a $40,000 college better than a $20,000 college, just as much as a $60,000 college is better than a $40,000 college? With domain knowledge, a user might be able to transform a continuous attribute to better match the truth. In this instance, maybe cost of attendance predicts good schools better when the cost is less than $50,000 but matters less for more expensive schools. The second method is problematic because the categories might arbitrarily separate objects; is a $39,000 college worse than a $41,000 college? Once again, a user with domain knowledge might be able to make better categories.

Both of the aforementioned encodings require a heavy amount of user knowledge to be informative. Often, however, there are not resources to acquire this knowledge, so the encoding must be simplified.

A second limitation to ranking is that the output does not provide qualitative information about the ranked objects. Take for instance, the top 3 colleges ranked by some model. The first two colleges might be almost identical and the third college drastically worse. A user would only see the enumerated list, and would not be privy to the quality of the colleges. From their perspective, College 1 is just as much better than College 2 as College 2 is of College 3.

Another limitation on ranking can be found in the "FA*IR: A Fair Top-k Ranking Algorithm" paper that claims to solve the Top-k Ranking problem (Zehlike et al., 2017).

2.2 Related Work

Presented is the list of tools that inspired the development process of RANKIT.

MATTERS

Massachusetts Technology, Talent, and Economic Reporting System (MATTERS) is "an online analytics dashboard empowered by a powerful dynamic data integration infrastructure" ("Matters" [APA], n.d., para. 1). It extracts and collects data metrics related to talent and economics from many public sources over time. And it allows users to explore, analyze and visualize these collected data by comparing cost, talent, and economic indicators across states.

MATTERS aggregates data from a variety of datasets related to talent, tax, business, quality of life into a single place. It then calculates the rankings of Talent, Tax, Cost of Doing Business, and Quality of Life for each state. The snapshot of MATTERS interface is shown on the figure below.

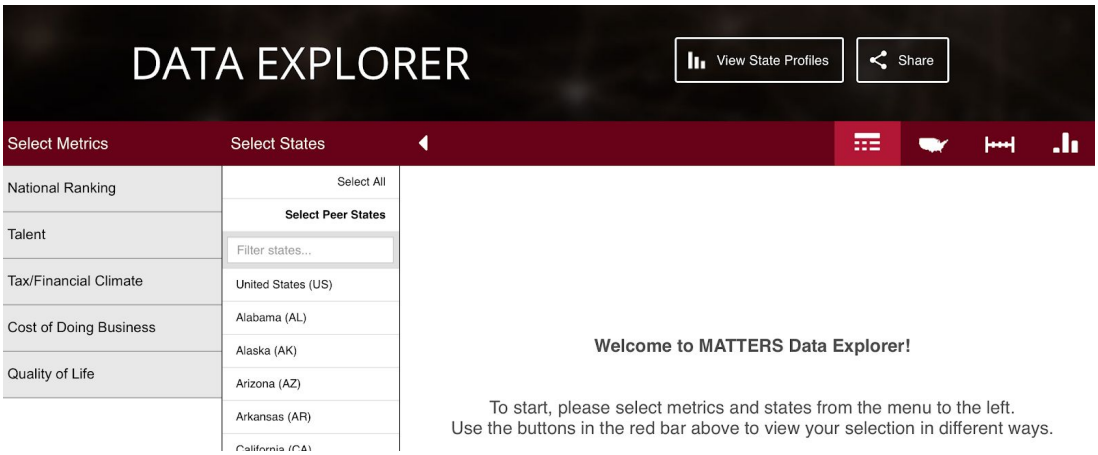


Figure 1. MATTERS Interface

Users are able to choose among different datasets and among different states to analyze how each state ranks among the rest. MATTERS provides four different visualization options: table view, heat maps, line charts, and bar charts.

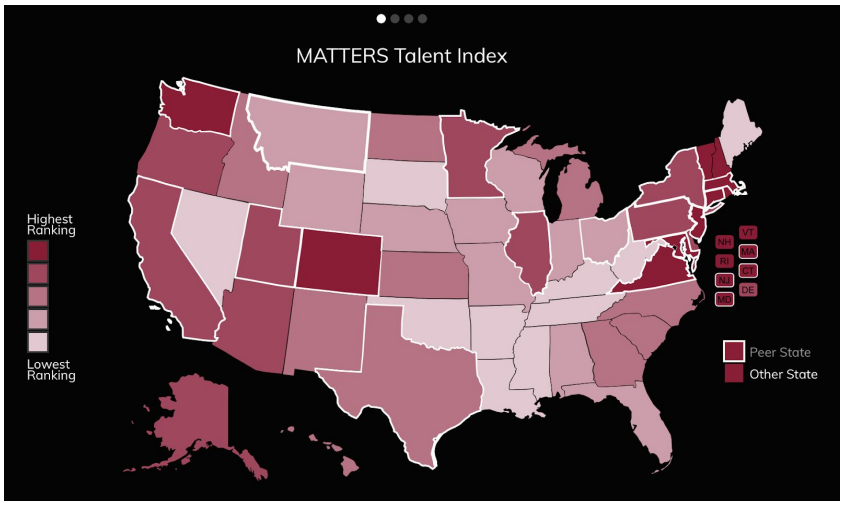


Figure 2. MATTERS Heat Map Visualization

Having a variety of datasets in a single place and visualizing the ranked data in different ways inspired the team to include multiple built-in datasets. The tool also influenced the ability to enable visualization of the data after constructing a ranking in the RANKIT system.

Lineup

LineUp is "an interactive technique designed to create, visualize and explore rankings of items based on a set of heterogeneous attributes" (Gratzl-Streit et al., 2013, section 19(12)). It enables users to control the complicated relationship between the attributes of a dataset. LineUp enables users to interactively combine different attributes to rank items with multiple attributes. It also enables users to change the weights of each attribute and visualise the impact of changes in the rankings interactively.

LineUp uses simple bar charts to represent the weights of each attribute in the process of ranking. Users can not only combine different attributes in the process of ranking, but also adjust the weights of attributes. Changes in ranking by adjusting the attribute combinations, or the weights of attributes are highlighted by integration of slope graphs.

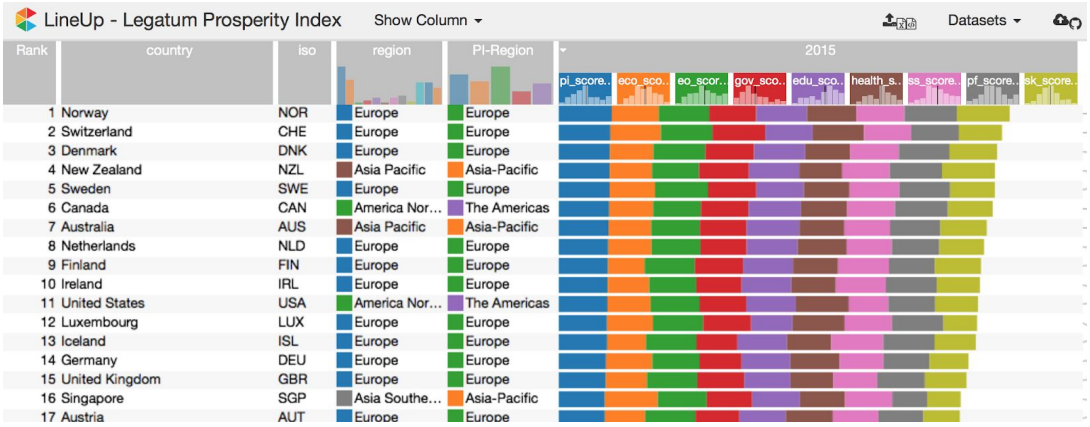


Figure 3. LineUp Interface

LineUp's ability to adjust the weights of each attribute in datasets inspired RANKIT to make customized rankings based on users' preferences. However, instead of requiring users to set the weights manually, users give a partial ranking by listing items (List Comparison), categorizing items (Categorical Comparison), or listing pairs of items (Pairwise Comparison), and RANKIT calculates the weights of each attribute based on users' preferences.

Podium

Podium is a tool that allows users to rank a subset of data and determine the weight for each attribute and the overall ranking of the data. The prototype system allows users to drag and drop rows in tables to rank order of data based on the users perception of the data value (Wall-Endert et al., 2013, pp. 288-297).

The key feature of this tool consists of an interface that allows users to visualize rankings through data tables and generate attribute weights from user interactions.

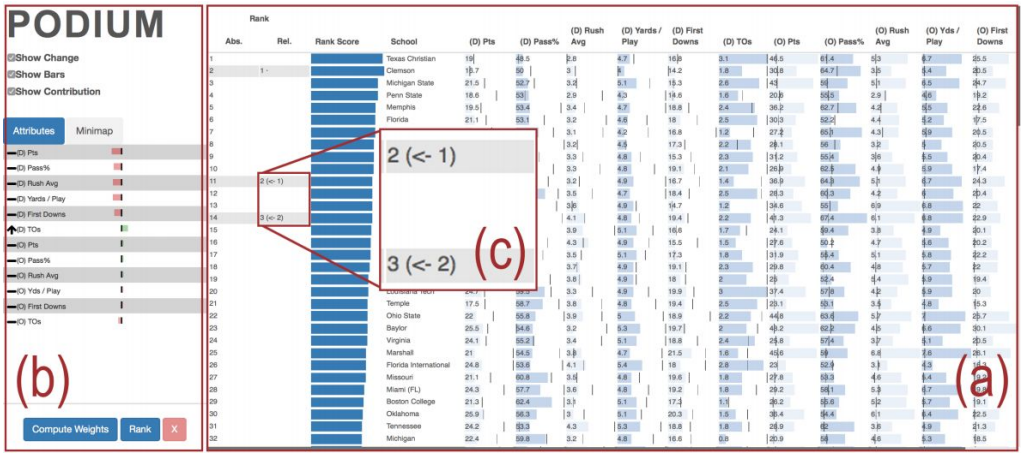


Figure 4. Podium Interface

Users who used Podium usually spent a short time using the tool resulting in small amounts of training data. As a result, the training data is not enough to produce accurate rankings as the complexity and number of objects increases. Learning from the flaws of Podium, RANKIT gives users the opportunity for a much more flexible approach by allowing them to construct their own rankings, using one of the three available comparison methods, while Podium uses just one comparison method.

3. Methodology

3.1 Overview

This section will describe different aspects of this project that were taken into consideration when choosing the development tools and a platform type. Additionally, it will cover how each of those options were evaluated.

3.1.1 Deciding an Application Type

Three application types were reviewed: Mobile, Desktop and Web. The advantages and disadvantages of each of them will be discussed in the following sections.

Mobile Application

As shown in Figure 5, the main advantages of mobile applications are their availability and ease of use for out of office users ("Chapter 20", 2009). However they have a limitation for input, navigation and screen display. Taking this limitation into account, the mobile type of application is not suitable for RANKIT's needs of interactivity and ability to collect the input from users.

Application type	Benefits	Considerations
Mobile applications	• **Support for handheld devices.** • **Availability and ease of use for out of office users.** • **Support for offline and occasionally-connected scenarios.**	• **Input and navigation limitations.** • **Limited screen display area.**

Figure 5. Advantages and Disadvantages of Mobile Applications

Desktop Application

Desktop applications offer some important benefits for the developer and the user. They remove the reliability on Internet connection, allowing users to keep using the application offline. They ensure a more secure storage of user data and allow for faster performance than web-based applications because they do not have browser overhead. Additionally, desktop applications require a minimal hosting cost for developers (Gnatyk, 2017).

One disadvantage of desktop applications is their lack of portability because users can only access them on a personal computer or laptop. It also requires an installation on each computer independently and a manual update on every single computer that uses the application. These aspects can make the experience more tedious for the user. Another disadvantage is an extra storage needed on the computer in order to store the application (Gnatyk, 2017).

Application type	Benefits	Considerations
Desktop applications	• **No reliance on the internet** • **Privacy for user** • **Fast** • **Low Hosting Cost**	• **Lack of portability** • **Installation required for user** • **Deployment** • **Extra storage needed for user**

Figure 6. Advantages and Disadvantages of Desktop Applications

Web Application

The advantage of web applications is their ease of use as they do not require an installation or regular upgrades. Additionally, they provide a platform independence due to their ability to run on any device with a browser. The web applications are adaptable to workload increase, since the developer can handle a greater workload with addition of the new servers to the system (Gnatyk, 2017).

The main drawback of web applications is that internet dependency slows down the performance of the application. It happens due to the data transmission in the Internet through HTTP requests and responses, therefore if the large amount data is sent and connection is not fast enough, the performance of the application will decrease (Gnatyk, 2017).

Application type	Benefits	Considerations
Web applications	● **No need to install** ● **No upgrades needed** ● **Platform independence** ● **Adaptable to workload increase** ● **Interactivity**	● **Internet dependency** ● **Slower performance** ● **Browser reliance**

Figure 7. Advantages and Disadvantages of Web Applications

Final Choice of Application Type

A web-based application type was chosen as the platform for this project. Out of the three types under consideration, the web application is the most suitable for the needs of the current project; to create an application that is both accessible for everyone and easy to interact with. As for the disadvantages of a web application, the slow performance is a very important issue. However given the size of the data utilized in this project, it is not enough to cause a drastic speed drop. In fact, currently the largest dataset takes on average from three to six seconds delay while loading in the Explore tool. Various algorithms were applied to make the computation rate faster and will be discussed in the "Implementation and Design" section of this paper.

3.1.2 Programming Language Choice

Nodejs

Node.js is a cross-platform framework for executing JavaScript code on the server side.
One of the biggest advantages of Node.js lies in its non-blocking IO system that provides an ability to process many requests concurrently. This allows the applications that are built with Node.js to achieve high scalability levels, hence performing faster (Why to Use Node.js, 2017).

On the other hand, Node.js does not support multithreading and does not perform long-running calculations due to the fact that the heavy computations block the incoming requests. This can lead to decrease of performance. In case of RANKIT, the server is responsible for executing machine learning scripts that require some heavy computations, which makes Node.js unsuitable for RANKIT's needs (Why to Use Node.js, 2017).

Python

Python is an interactive, object-oriented, high-level programming language. It provides effective tools for data analysis and has libraries for machine learning. These two aspects are very important in RANKIT, since the tool employs machine learning. Additionally, it gives an opportunity to make heavy computations on the server side with a low impact on the application's performance.

However, one disadvantage is that Python has a low latency and slow performance in communication (Krill, 2015).

Hybrid Model of Python and JavaScript

The hybrid model represents Python as the server-side language and JavaScript as the client-side environment. This approach allows developers to use web frameworks and environments to be more productive and efficient in Python. Additionally, this approach provides the external libraries to set up the communication between the JavaScript client and the Python server. Nevertheless, Python's slow communication with the frontend is one of the main problems when loading larger-sized data.

While developing RANKIT, Flask was used as a web framework on the backend, utilizing Jinja2 templates for transferring the JSON-formatted data from the Python server to the JavaScript client. This method gives an extreme flexibility and ease in URL routing, as well as allowing for rapid deployment and development. Since RANKIT requires intensive machine learning calculations to build a personalized ranking, Python is important to use because it supports both the machine learning built-in libraries and heavy computations.

3.2 Early Prototypes

One of the main goals of this project is to create a tool for individuals both knowledgeable and not when ranking a subject of interest. The application went through series of design and development iterations, focusing on making the user's interaction with the tool maximally intuitive and effective.

The first prototype of the landing page was designed to be informative, yet clean and compelling for the user. The prototype is shown in Figure 8.

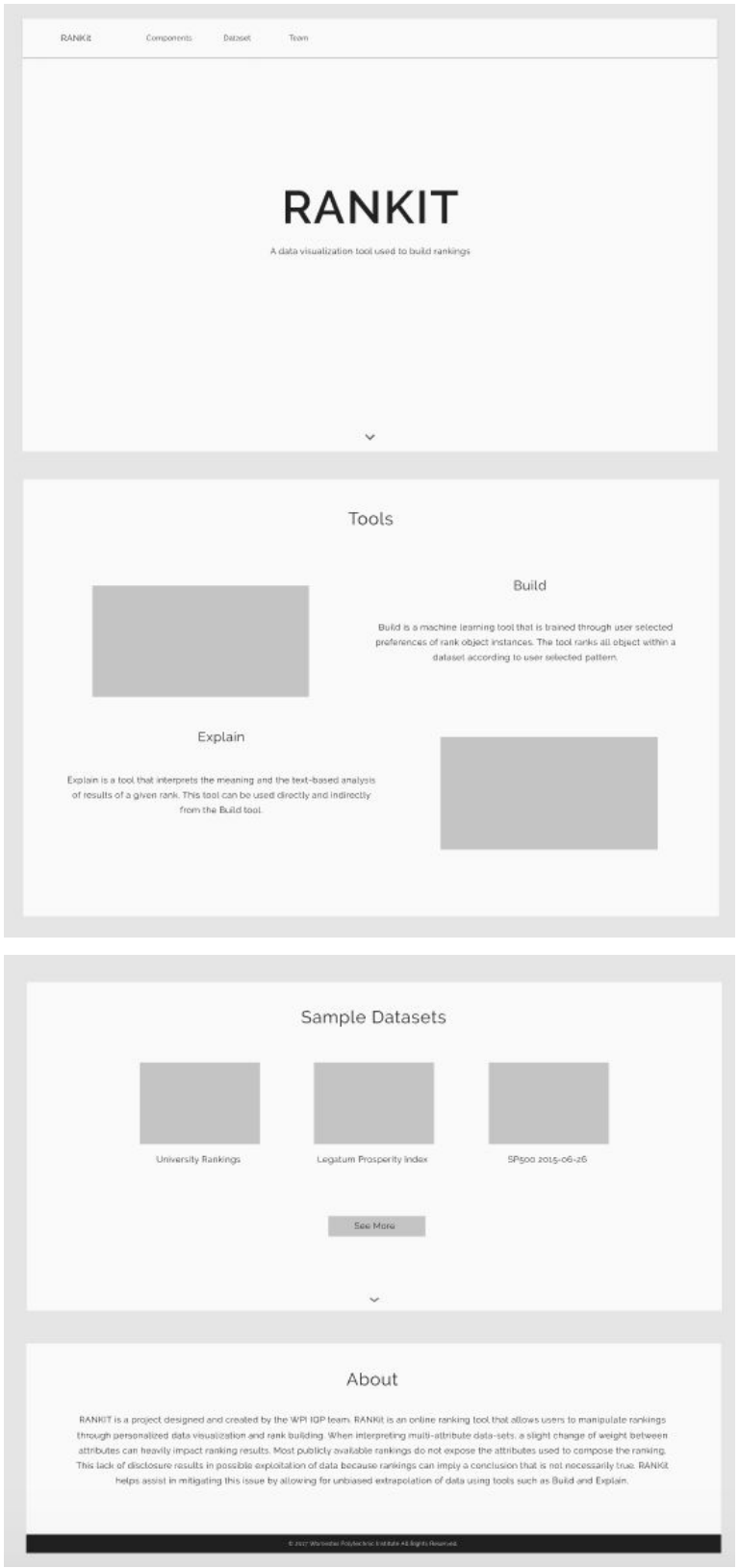


Figure 8. A Prototype of Landing Page

During the first iteration, an informal user study was conducted to investigate humans' natural thought process and perception of rankings. The names of US colleges were printed on small cards and were placed in a shuffled manner on the table. The subjects (n=5) were randomly picked and asked to choose some colleges from the pool of cards on the table, and rank them according to their own preferences and domains of knowledge. It is important to note that the way the subjects could have organized their ranking was intentionally not specified.

The results of this informal study showed that there were three main methods that people intuitively use when asked to rank data. Two out of five participants used a "List Comparison" method where they organize their colleges in a sorted list with the most preferred objects on top and the least preferred objects on the bottom. One participant used the "Categorical Comparison" method where they classified all of their chosen colleges into three categories: high preference, medium preference and low preference. One subject used a combination of "List Comparison" and "Categorical Comparison" methods by sorting the objects in each category of his preference. Finally, one of five participants used a "Pairwise Comparison" method where they organized colleges in pairs, giving a better rating to one of the colleges in each pair.

Based on the results of this informal onsite survey, three main comparison methods of ranking were revealed and used to design the "Build" component of RANKIT. Figure 9, Figure 10 and Figure 11 demonstrate the early prototypes of each of the comparison methods of "Build" component.

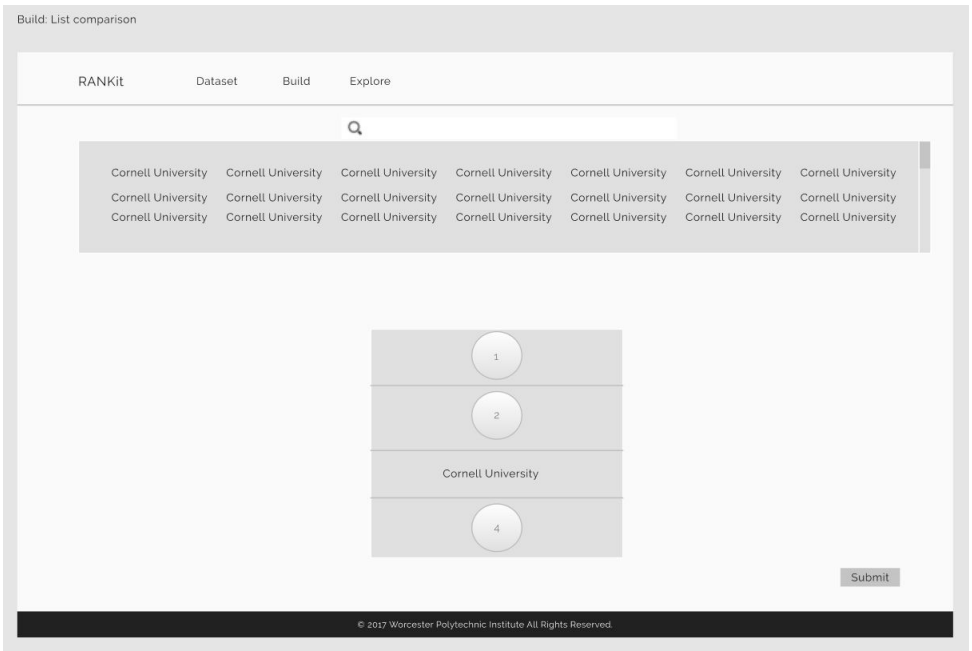


Figure 9. List Comparison Prototype of Build Component

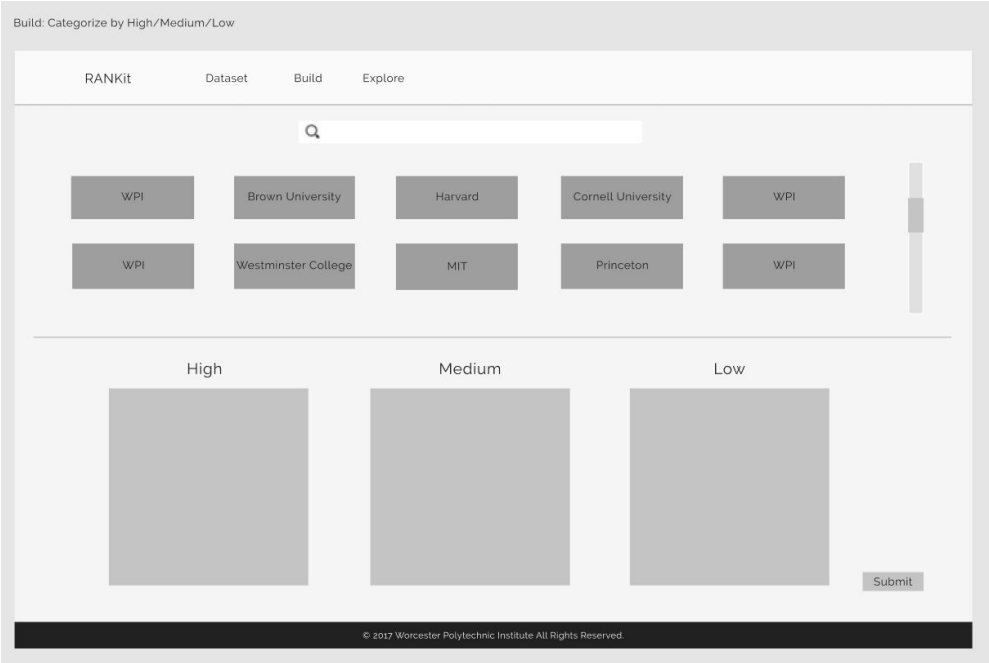


Figure 10. Categorical Comparison Prototype of Build Component

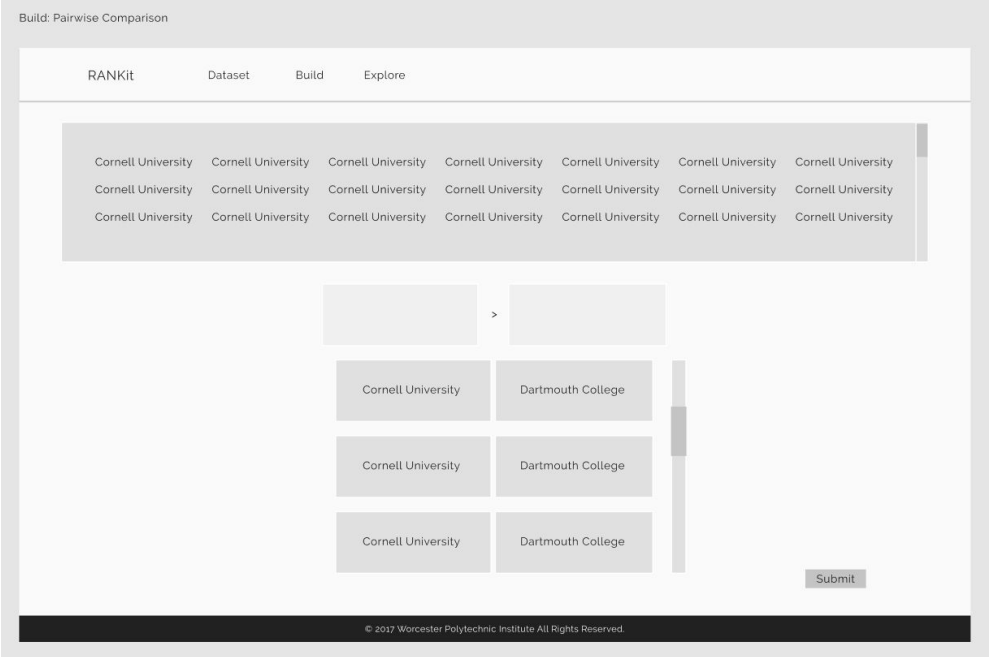


Figure 11. Pairwise Comparison Prototype of Build Component

Additionally, one of the objectives of the current tool is to have valuable datasets that can offer more options and flexibility to the user. The prototype for a separate page containing all the available datasets was designed and is shown on Figure 12.

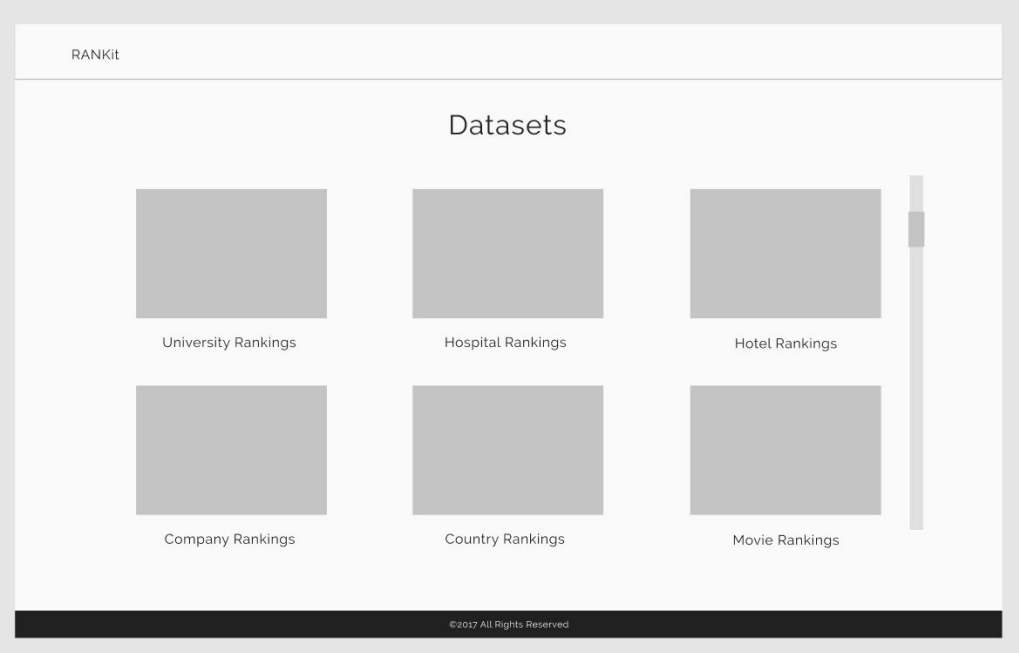


Figure 12. A Prototype of Dataset Selection Page

3.3 Design and Implementation

3.3.1 The Server

Directory Structure

Reducing a large project into a smaller and reusable application allows for easier addition of new features that are more maintainable and easier to identify errors with. Flask provides a method, called Blueprints, to separate distinct sections of the high level website structure and the code's function.

There are three important parts of a Python project: template files, static files and view files. Template files are the raw HTML files. Static files contain images, CSS styles, javascript functions, and videos. The view files are responsible for routing information, by identifying an endpoint and choosing the appropriate template file to load.

There are two ways to structure directories: functional and divisional. For a functional structure, each part of the project is grouped together. Templates, static files and views are separated into directories, with a subset in each per Blueprint.

rankit/

```
static/
templates/
    home/
    build/
    explore/
views/
    home.py
    build.py
    explore.py
```

The divisional structure establishes greater separation between the blueprints. The parts of the project correspond to the Blueprint they belong to, such that all templates, static files and views are found under the specific Blueprint.

```
rankit/
    home/
        views.py
        templates/
        static/
    build/
        views.py
        templates/
        static/
    explore/
        __init__.py
        views.py
        templates/
        static/
```

For projects with tighter coupling in between Blueprints, the functional structure is ideal as it allows components to interact and share general layouts. For projects with looser coupling, divisional structure is beneficial. RANKIT uses divisional structure because the tool consists of two independent projects (Build, and Explore), which, can and are encouraged to work together, do not depend on one another and can be used separately. Furthermore, the expansion of RANKIT would be simplified if new tools can be added and old ones removed independently from one another.

Communication between the Client and Server

Three JSON description files are used: one for loading data to the Build Tool, another for sending user preferences from the web client to the Build Tool, and finally one for sending ranked objects to the Explore tool after the machine learning script ranks from a user's partial ranking.


```
[
   "Big Hero 6",
   "Minions",
   "Deadpool",
   "Guardians of the Galaxy",
   "Jurassic World"
]
```


Figure 13. JSON file for Communicating Dataset Objects from the Server to the Web View

It is important not to expose more information than is necessary for analysis. In the context of ranking by object, it is assumed that the user has some previous knowledge or intuition about the object as a whole, not attributes that come to define them. As shown in Figure 13, the file that is sent from the server to the web page consists only of object identifications.


```
[
   {
      "high":"Minions",
      "low":"Deadpool"
   },
   {
      "high":"Guardians of the Galaxy",
      "low":"Jurassic World"
   },
   {
      "high":"Big Hero 6",
      "low":"Minions"
   }
]
```


Figure 14. JSON file for Communicating User Intent from the Web View to the Build Tool

However, when looking at data gathered from the user, it is important to know three things: the two objects being compared, and the preferences of one over the other. This is done so by referring that an object under the “high” key is prefered over the object under the “low” key. In the instance of Figure 14, the user prefers “Big Hero 6” over “Minions”. However, “Minions” are prefered over “Deadpool”.

The last JSON consists of the whole dataset and the final ranking. Along with the rankings, the attributes and their weights are also sent to the frontend to ensure that the application is transparent in constructing its ranking.

Retrieving Partial Rankings

Originally, the backend of the Explore tool was only accessible through a POST request. This led to a major problem where, upon refreshing a page, users lost their desired ranking. Ultimately, users should be able to re-access their computed scores even after some time. Therefore, the backend was refactored to manipulate the url to access the rankings using a GET request. Such that, by accessing the specific URL the user can go back to their previous ranking in the Explore tool at anytime.

The Explore tool URL is in the form of “<number_of_pair> = <high_pair> > <low_pair> &”

An example URL is /explore/states/0=Arizona>Colorado&1=Florida>Arizona&

In such an example, Arizona was ranked higher than Colorado, with another pair where Florida is greater than Arizona. Parsing the url is a regular expression:
$\backslash d +=([\backslash w \backslash s' -:!\backslash., ()] * [\backslash >]\{1\}[\backslash w \backslash s' -:!\backslash., ()] *)\&$

The regular expression matches and extracts pairs within a “number= … &” formulation. Because of such a parsing, an item name within a dataset cannot have the following symbols: = & >. However symbols such as - : ! . , ( ) are allowed.

Machine Learning Script

There are multiple approaches to learning to rank: pointwise, pairwise, and listwise. The ranking problem in a pointwise approach is reduced to a classification, regression, or ordinal classification problem, where the label of an object is predicted given that object (Li, 2011). In a pairwise approach, the ranking problem is reduced to a pairwise classification or a pairwise regression. Instead of reading in points and learning their order, the difference between pairs of objects is measured and that difference becomes a vector that will be projected in a new space. Then, a classifier against the new points is learned and labels are assigned to them. The decision boundary is set, and the difference between points and the decision boundary becomes the

ranking. The listwise approach takes in the group structure of ranking for both learning and predictions (Li, 2011).

There exist multiple algorithms to learn ranking functions. RankNet uses a neural network and gradient descent methods to model a ranking function (Burges et al., 2005). A semi-supervised learning to rank algorithm, SSLambdaRank, optimizes accuracy by grouping similar items to have similar preferences to each one another (Szummer & Yilmaz, 2011). Ordinal regression, establishes an equivalence relationship on pairs of objects (Herbrich, Graepel, & Obermayer, 1999). RANKIT builds on the MyRanker paradigm, but with a visual interface, and uses the pairwise approach where pairs of data are evaluated, thereby translating the learn to rank to simple classifications (Kuhlman & Rundensteiner, 2017). Regularized-least squares algorithm is employed as the classification algorithm. The input to the algorithm is a list of pairs and a label indicating which object in a pair is preferred over the other. The pairs, containing partial ranking information, are provided by the use. Those pairs become a training data that formulates a model, which is a prediction over the entire dataset. The machine learning script in RANKIT utilizes the RankRLS library, which uses a variation of the regularized-least squares algorithm called RankRLS (“Learning to rank”, 2016). This machine learning algorithm learns the relationships between objects by analyzing pairwise comparisons to build a full ranking. RankRLS minimizes the error function of the general least-squares approximation algorithm (Pahikkala et al., 2007).

3.3.2 The Client

Throughout the project, providing an intuitive design of the user interface was an important factor. The goal was to ensure that users understand the purpose and functionality of the web application as soon as they enter the landing page. To accomplish this, the team did some research on common practices of website design and structure while concurrently developing on a working prototype.

Intuitiveness of the User Interface

Originally, the main design prototype of the landing page had plenty of information on what the project was about, potential use cases, instructions, dataset selections, and much more. This was quickly changed to improve navigation efficiency. Without efficient and user-friendly navigation, the user is likely to get confused and lose interest in using a website (Gehrke & Turban , 1999). In the prototypes case, many users could not comprehend all of the landing page due to its large amount of textual information. Therefore, it was important to keep in mind users’ involvement because a user who does not understand the purpose within the common human attention span time is less likely to continue using the application (Stone-Minocha et al., 2005).

In order to solve this issue, the team used an approach to filter out content that would not be necessary for a new user when first familiarizing themselves with the web application and relocated it to other areas of the tool. For example, the project's history and team information has been relocated to a separate about page where users can learn more about the history of the project's development. Specific instructions on how to use the Build and Explore tools were relocated from the landing page to a help button located in the upper right corner in both of the respective tool's pages.

The styling for the application was pulled from Twitter Bootstrap Cascading Style Sheets files as well as javascript functions made from scratch to indicate valid submissions. Using Bootstrap allowed the website to be both responsive and consistent in style throughout the entire tool. Originally, all buttons in RANKIT were animated with different transitions and colors that were pleasing to the eye but detracted from the goal of project consistency and design philosophies. Therefore, to avoid obtrusive buttons, each button was redesigned to be consistent in sizing and color scheme.

The "Rank" button located in the Build tool is the exception. This is because javascript functions validate ranked arguments. In other words, the script disables the rank button preventing the user from ranking if no items were entered into the desired ranking method. Each ranking method has its own rule by which the button is enabled. In List Comparison, the "Rank" button is active when at least two items have been ranked. In Categorical Comparison, the "Rank" button is active when there exists at least one item in at least two different categories. In Pairwise Comparison, the "Rank" button is active only when all pairs are complete. Color is used to indicate the status of the ranking methods. A grey button is disabled, and it transitions to red when the case arguments are met, as shown in Figure 15.

Figure 15. Script Checks for Valid Arguments and Reflects Success Through Color Transition.

To help guide the user to use the drag and drop feature in the dataset item selection section, the team added a white outline and a cursor change when hovering over each item. For the landing page, the animated color hue transitions were achieved through an array of hex color codes that animate linear gradients. This grabs the attention of the user and introduces them to the website's color scheme (Stone-Minocha et al., 2005).

Landing Page

The team made several design iterations throughout the project's development cycle using feedback on the previous versions. The landing page was designed to provide enough information for new users to get started as well as allowing experienced users to quickly skip over the introductory steps. The first screen seen when a user enters the landing page is an animated title of the web application named "RANKIT" (shown in Figure 16) with a short description below the title explaining what the tool accomplishes.

On the top right of the screen are two options, one called "About" for the about page and another called "Start Ranking". The former link takes the user to the Dataset Selection page without any instructions, targeting users who have used the tool before. The design choice of placing a "Start Ranking" link addresses the previous issue where returning users had to follow through the tutorial section before being able to access the tool. The "Learn More" button scrolls the webpage down to the instructions section below, forcing non-experienced users to first understand how to use the tool. To recap, the "Start Ranking" button and the "Learn More" button both serve the purpose of bringing the user to the Dataset Selection page but differ in flow of scrolling movement.

The information is organized into three columns explaining a step-by-step process: the options for selecting a dataset and the tools to view the selected dataset. The about page is included for users who are interested in learning more about the purpose of the project, the development team, and contact information for feedback.

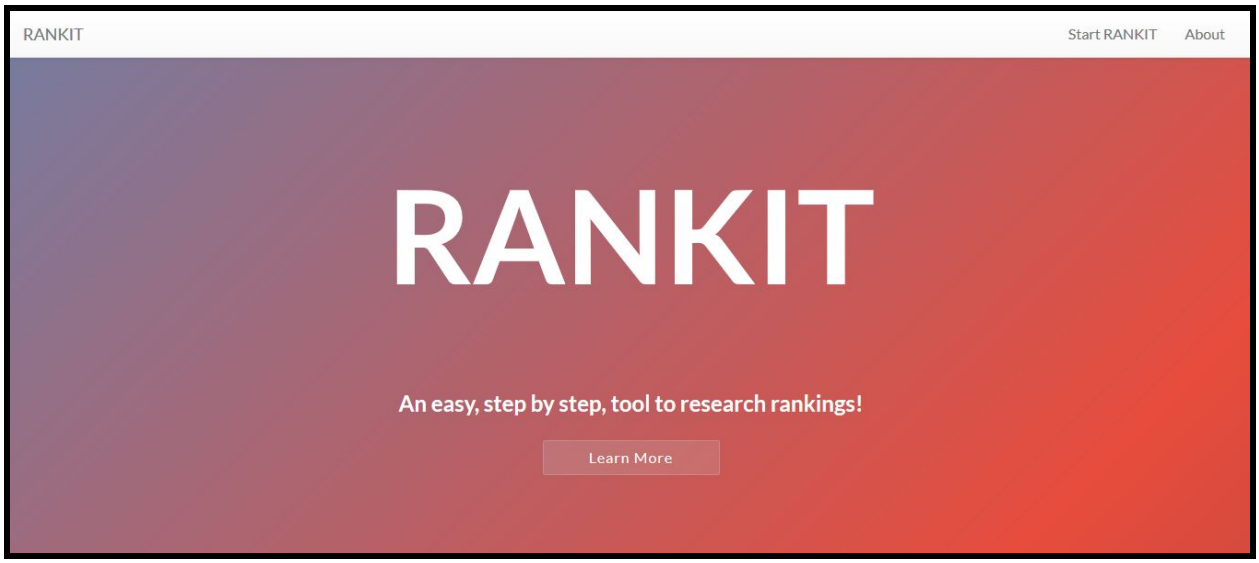


Figure 16. Main Landing Page

Dataset Selection Page

Selecting either of the "Start RANKIT" buttons will bring the user to the dataset selection page. For the user study, a selection of six datasets were provided. These datasets include States

("States Dataset" [APA], n.d.), Colleges ("2018 Best Colleges" [APA], n.d.), Movies ("Movies Datasets" [APA], n.d.), FIFA ("FIFA Datasets" [APA], n.d.), Board Games ("Board Games Datasets" [APA], n.d.), and Video Games ("Video Games Datasets" [APA], n.d.). The selection of these datasets were based on the meaningfulness of the attributes found in the datasets. To make the selection process easier for users to distinguish, each datasets is represented with its own background photo. Each dataset option is also set to a grid format allowing users to easily navigate through the screen. Originally, the dataset section would allow users to select a dataset and then have the options, Build and Explore, appear in an animated transition. However, in order to mitigate the issue where touch screen devices did not displaying the transitions, this styling was reworked.

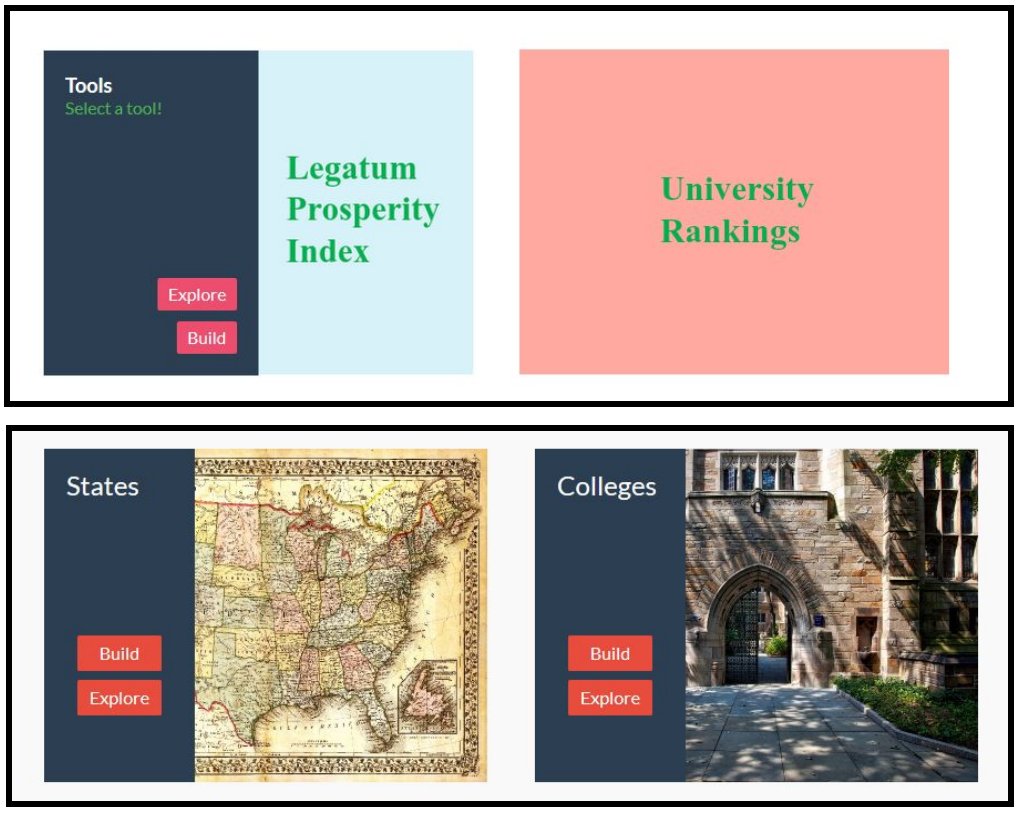


Figure 17. Iteration Differences for Dataset Selection Interface. The top image represents the original interface while the bottom image represents the final design changes.

Selecting a desired dataset and clicking on the Build tool triggers a popup on the user's screen, providing the option for selecting a ranking method. Hovering over each method with a cursor reveals a description about the method type. Earlier versions of the project forced users to go straight to the Build tool's List Comparison method. However, this led to a bias favoring the default method. This bias influenced the team to have the process redesigned such that the user

could select the rank method of their choice first before being taken to the Build tool. It also allowed the user to familiarize with each method before moving onto the next step of ranking.

Figure 18. Pop Up Menu. Allowing the user to get acquainted with the ranking methods before they enter the Build Tool.

Build Tool Page

The Build tool page went through many different design iterations, focusing on the how users rank items of a dataset. During the first iteration, the design contained two sections where a user would drag individual items from their selected dataset, and drop into a rankable box. The section of items was located in the top of the screen while the ranking method section was located on the bottom of the screen. However, this process proved to be tedious for users. Users had to scroll to the top to first select a desired item, to later scroll down to rank the item, repeating the same process another time through. This process ended up disrupting the flow of selecting items. For the final design, the sections were moved such that the database item selection was placed on the left and the ranking methods section was placed on the right. User feedback indicated that it was much easier to drag and drop an item from left to right because it negates the need to scroll from top to bottom, and instead showed the entire Build tool on the screen.

In the Build tool, users have the option to search for desired items, sort them by alphabetical order, or shuffle them randomly, as shown in Figure 19. The shuffle feature was implemented with intention to remove biased item selections that were made based on what the users' first viewed on the page. Based on user's feedback, people tend to pick the first items they see on the page, creating biased rankings formed from a biased item selection. Shuffle allows users to randomize different items in the dataset section, providing a better method of sorting and mitigating biased ranking selections. Users can also revert Shuffle by using Sort and thus bring the item organization back to the default alphabetical order.

Figure 19. Build Tool Querying Options: Search, Sort, and Shuffle

The build tool has three different methods for users to input partial rankings: List Comparison, Categorical Comparison and Pairwise Comparison. Since the ranking script works only on a list of pairwise data, to adhere to the format outputs from both List Comparison and Categorical Comparison are converted to the output of Pairwise Comparison.

In List Comparison, users can input a linear list of items in the order of decreasing preference. Suppose a user inputs a list of A, B, and C. In the Pairwise Comparison, it is equivalent to -

- User prefers A over B.
- User prefers A over C.
- User prefers B over C.

So generally, for $[O_1, O_2, \dots, O_n]$ where O_i means i^{th} item in the list and $n > 0$, there are $(n^2-n)/2$ numbers of pairs generated. This method reduces the amount of data users would input in a Pairwise Comparison method to an order of 2.

In Categorical Comparison, users have to input items in three discrete buckets: high, medium, and low preference. One fundamental distinction between List Comparison and Categorical Comparison is that in Categorical Comparison, the order of items within each group does not imply preference. Suppose a user inputs A and B in high preference; P and Q in medium preference; X and Y in low preference. In the pairwise level, it is equivalent to -

- User prefers A over P
- User prefers A over Q
- User prefers A over X
- User prefers A over Y
- User prefers B over P
- User prefers B over Q
- User prefers B over X
- User prefers B over Y
- User prefers P over X
- User prefers P over Y
- User prefers Q over X
- User prefers Q over Y

So generally, users input partial ranking over a set of items $\{O_1, O_2, \dots, O_n\}$ but not for every (O_i, O_{i+1}) pair. Therefore, Categorical Comparison is a compromise between the succinctness of List Comparison and the flexibility of Pairwise Comparison.

In Pairwise Comparison, users explicitly input a list of pair. This approach gives users full control of comparing items in exchange for succinctness of the input.

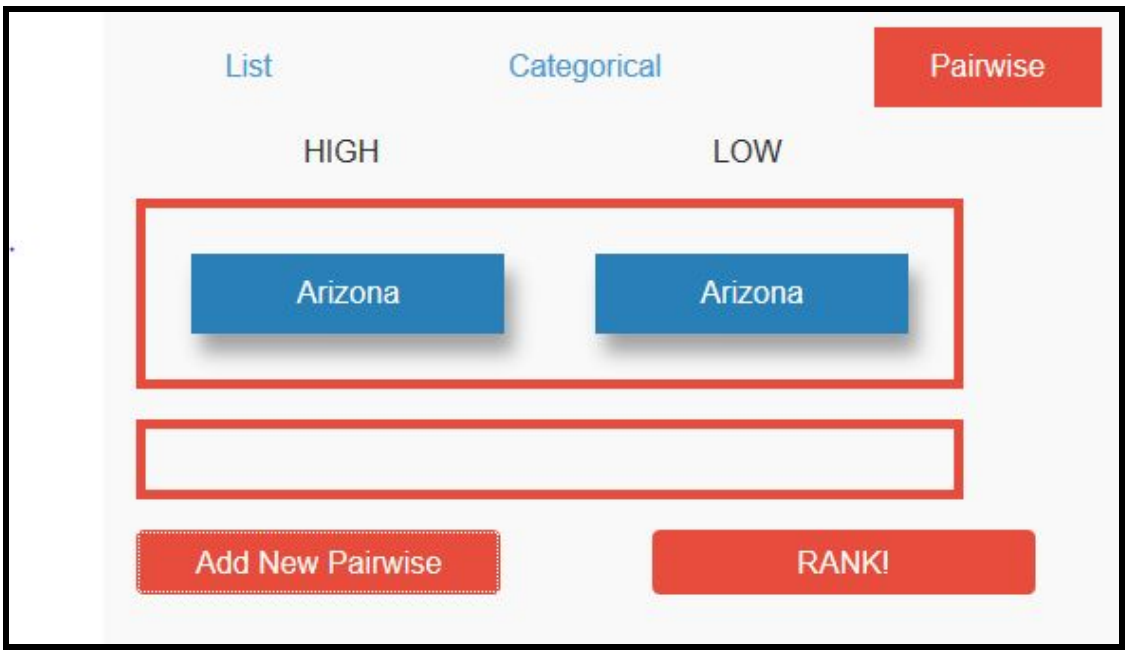
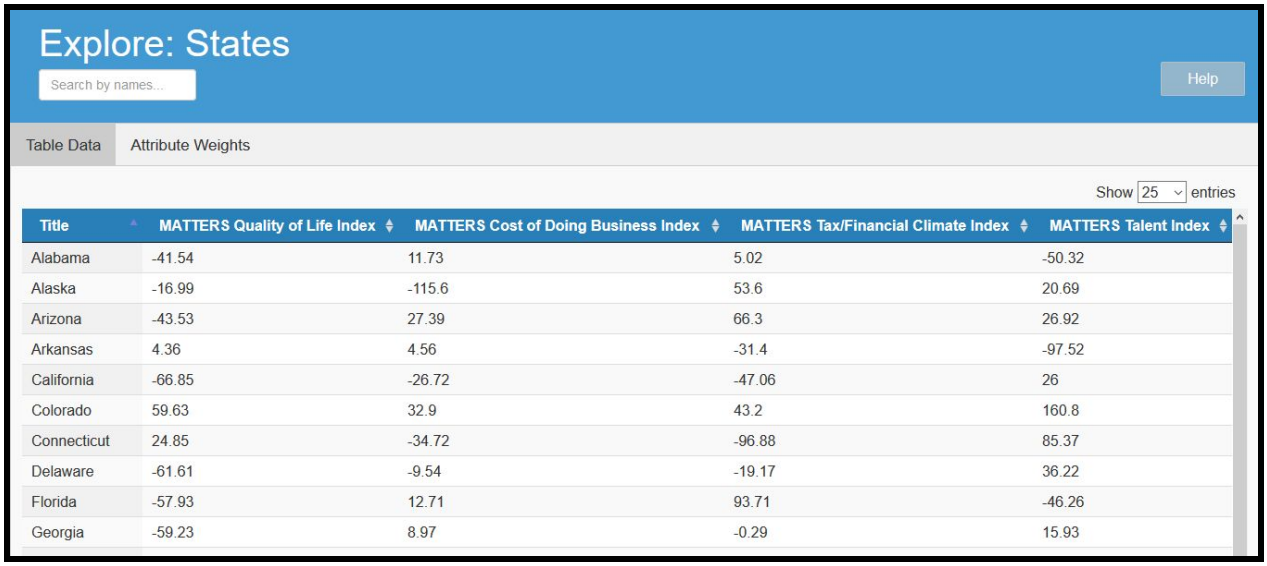


Figure 20. Pairwise Method Interface

Explore Tool Page

Users can view and browse dataset information using the Explore component. The tool can be used without first building a ranking, but only allows users to view weighted attributes after ranking has been made. The initial draft iteration only had Tableview as a feature displaying the subset impact of the rankings compared to the overall dataset (shown in Figure 21). However, users were having difficulty understanding the meaning behind their ranking. To address the issue, the team added a weighted attributes section that displays attributes and their relative importance to the overall effect on the dataset's ranking (shown in Figure 22).



Title	MATTERS Quality of Life Index	MATTERS Cost of Doing Business Index	MATTERS Tax/Financial Climate Index	MATTERS Talent Index
Alabama	-41.54	11.73	5.02	-50.32
Alaska	-16.99	-115.6	53.6	20.69
Arizona	-43.53	27.39	66.3	26.92
Arkansas	4.36	4.56	-31.4	-97.52
California	-66.85	-26.72	-47.06	26
Colorado	59.63	32.9	43.2	160.8
Connecticut	24.85	-34.72	-96.88	85.37
Delaware	-61.61	-9.54	-19.17	36.22
Florida	-57.93	12.71	93.71	-46.26
Georgia	-59.23	8.97	-0.29	15.93

Figure 21. Explore Table View. Explore Tool displays of each attribute based on the user's ranking

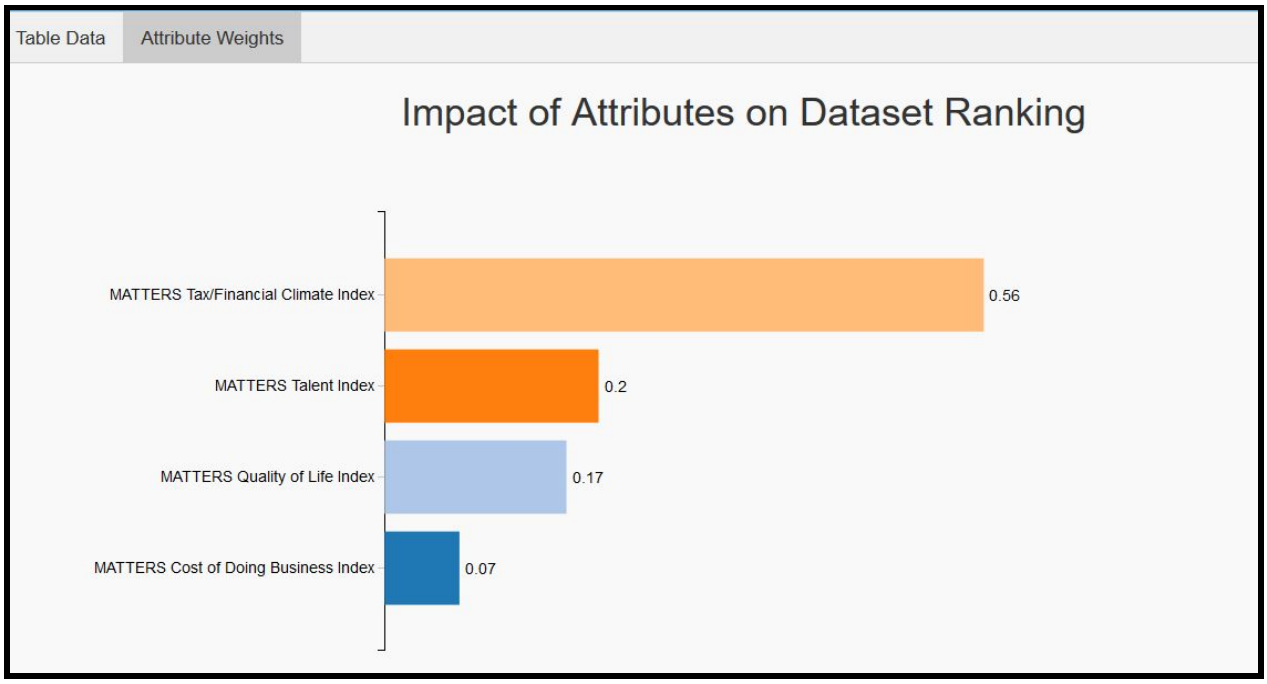


Figure 22. Weight of Each Attribute. The Explore Tool displays weights based on the user's Ranking

3.4 Team Structure

Team Structure

The team is composed of four undergraduate students, Goutham Deva, Diana Doherty Malika Nurbekova and Zarni Phyo, working with two mentors: Professor Elke Rundensteiner and a graduate student, Caitlin Kuhlman. MaryAnn VanValkenburg, an undergraduate student, is working closely with Caitlin advising the team about ongoing research for the ranking tools. During the development team meeting, the product manager, Diana, is responsible for creating and assigning weekly tasks to everyone on the team, checking up on progress, taking note of any blockers and re-allocating work if necessary. The team also has a note taker, Goutham, who notes down key details discussed, sending a summary of weekly meetings to everyone. The team is divided into backend and frontend specialists. While the beginning of the project focused on both ends of the developing stack, and the division was more apparent in the earlier part of the project, in the later half, the backend developers switched to work on the frontend as well.

Meeting Details

The full research team meets once a week for an hour. During these meetings, the team discusses what they have accomplished, gets feedback from the mentors, and finalizes tasks for next week. For a separate hour in the week, the team meets without the professor to inform the graduate student mentor of any blockers, to ask questions, and to work together on any parts of the weekly tasks. Meeting notes from these sessions are later discussed by the developing team and actionable items are produced.

The development team meets three to four times a week for at least an hour per meeting outside of the meetings with the mentors. During these meetings, the team either works together on the weekly tasks, discusses goals, prepares for the meetings with the mentors, or communicates encountered problems.

Originally, weekly tasks were organized and assigned on Trello. However, that proved to be ineffective because the future assignments cluttered the board of unnecessary for the week tasks. Therefore, the team switched to keeping track of everyone’s work on a powerpoint, referring to it when discussing progress. To not forget long term tasks that could not fit in for the week, the product manager kept a seperate list of tasks to accomplish in the later weeks.

4. User Study Overview

4.1 Goals

With the problems of ranking in mind and a tool at hand, it became important to evaluate the success of this tool in the context of problems in ranking. Since RANKIT is a web application designed to help users customize and interpret rankings to meet their own end goals, the purpose of the user study became to evaluate the usability and functionality of the application and to get feedback for improvement. Thus the following goals were constructed:

- Evaluate which method of building a ranking is most favorable among the three presented: List, Categorical, Pairwise.
- Estimate an amount of partial information the user is willing to input to achieve better results.
- Determine the intuitiveness of the user interface.

While the core of the machine learning application functions on the essence of Pairwise Comparisons, it is important to note the usefulness of abstracting the computation to the user. More precisely, for the common user, it might be easier to rank by a method that is more intuitive to them and, therefore, not feel alienated by the underlying mechanism. In feeling comfortable with the tool and its flexibility, the user should feel as though the tool has a purpose and can be applied to their lives. To draw this conclusion and determine whether the abstraction is necessary, it is important to note whether there exists a ranking method that is prefered by the majority and whether that method is Pairwise Comparison or not.

Typically users do not want to provide more work than necessary. A tool might be useful and accurate, but if it takes an hour to start up, it is very unlikely to be used. However, with little guidance, the tool cannot provide satisfactory results. It is thus important to find the equilibrium of user work and accuracy of results. From then on, the tool can be evaluated at that threshold to determine whether it is satisfactory to user needs.

Lastly, a great factor in tool usage is its usability. While the concept might be appealing to individuals, if it cannot deliver its purpose in a clean and usable manner, it might not be usable enough. It must thus be tested whether the tool is user friendly and intuitive for an average user to dive into and explore.

4.2 Overview

The target audience was people who have an interest in ranking data. Previous experience or specific knowledge of ranking is not required. The user study tested whether the tool can be accessible for those knowledgeable in the area of ranking and those new to the experience. The user study focused on individuals between the ages of 18 to 50.

People in different professional fields and age categories were targeted:

- Undergraduate/Graduate students of different majors
- Faculty from computer science, business, economics and data science department
- Experts in human-computer interaction
- People off-campus

Subjects were divided into two groups. The first group partook in an online survey and the second will be part of an in depth interview session conducted by one or more members of the RANKIT team. Each of the evaluations will have practical tasks that will consist of hands on use of the web tool as well as a written survey targeting the experience.

4.3 Onsite Interview

The onsite interview subjects were divided into two participant groups: experts in human-computer interaction (n=3) and students (n=10). The goal for each of the onsite interview groups was to evaluate the intuitiveness of the RANKIT application and the degree of users' understanding of the purpose of the tool.

The questionnaires for both groups contained identical questions. The interviews were conducted with one participant and two interviewers per session. Each subject was given a consent form to sign and was provided with a device that had the RANKIT application up and running. The interviewers navigated a participant to the landing page and started providing instructions to the subject.

The whole interview process was divided into two tasks. For the first task, interviewer one gave the exact instructions to the participant for each action under test including the dataset selection and partial ranking input, while interviewer two watched the process and took notes about the participant's experience with the application.

The second task gave the freedom for the participant to choose his/her preferred dataset and perform the partial input of any objects in any amount. Both interviewers observed the user's interaction and took notes about any comments or feedback the participant provided.

Finally, the first interviewer asked the subject questions from the questionnaire and the second interviewer recorded the answers. The onsite interview was considered complete once the subject answered all the questions from the questionnaire.

The responses from the experts in human-computer interaction were expected to be qualitative, more in depth and detail-oriented, and used for improvement in UI and UX. The responses from the students group were expected to be more quantitative, and targeted on different features, their functionality and the application's entertaining aspects.

4.4 Online Survey

While the interviews involved a more in depth questionnaire, the same tactic did not apply for the online survey. Upon its creation, the original questionnaire aimed to apply to both the in-person interviews and the online surveys. The questionnaire consisted of twenty three questions, taking approximately thirty to fifty minutes to complete, each of which contributed to testing the goals. However, a testing of the questionnaire in a form of an online survey revealed an underlying problem. While the interviews involved constant encouragement and guidance of the interviewer, the online survey did not have that ability. The lack of reinforcement from the interviewer resulted in users whose attention span did not last through the lengthy process of ranking with one method, switching to another and repeating. It became apparent that there were too many questions to ask in a survey and still retain interest. Since there were no direct benefits to participate in this study, and the good will of contributing to the creation of an intuitive and effective ranking application has different limits, different solutions were explored.

In wanting to give the interviewees incentive to complete the tasks and the accompanying questionnaire, mechanical turk was considered as a surveying platform. Mechanical turk is a crowdsourcing marketplace where workers perform tasks, such as questionnaires, and get paid at a per task or question basis. Choosing mechanical turk would ensure a secure and fast way of gathering participants for the survey. Before declaring mechanical turk as the final platform, cost to conduct the survey needed to be calculated.

From researching the number of responses an online questionnaire should aim to have, it was concluded that one hundred individuals was a good sample size. After reviewing the typical pay rate per question, a payment of 15 cents per minute was concluded to be the expected value. Since the total of twenty three questions, took approximately thirty to fifty minutes to complete (making it about 2 minutes per question), the following calculations were made.

Cost per question: 30 cents

Number of questions per survey: 23
Total interviewees: 100
The final calculations per cost: $690

Despite the advantage of mechanical turk, the survey was too long to gather important funding for the required payment, and thus the questionnaire itself was reevaluated. In reducing the need for interviewers to partake in conducting the study, the online survey gave way for faster evaluation of the RANKIT tool. Therefore, the focus of the survey transitioned from being a different way to test the same questions as the in-person interview to expanding the responses and reducing the time it took to take the survey.

The outcome of restructuring the online survey to be a quick and easy resulted in three surveys. Each survey differed only in the method of ranking. Since most of the interview questionnaire involved exploration of the Build and Explore tools, and relied heavily on letting the user discover and understand the meaning of the application, the online survey striped out the uncertainty of the interviewee getting stuck and supplied all of the important information. To not require the need for one specific interviewee to explore all Build methods (Goal 1), they will just be given access to only one. Results on the overall application would then be reviewed depending on which of the three methods interviewees were asked to rank by and whether that affects the overall satisfaction of RANKIT.

The final design of the interviews consisted of a one-time session where the participants were asked to perform six tasks using the ranking application, with ten questions total. After performing each task by exploring the web application without guidance, they were asked to rate their overall quality of the interaction using a questionnaire.

5. Evaluation

5.1 Onsite Interview

The following section describes the results of the onsite interviews for both study groups: experts in human-computer interaction (n=3) and students (n=10). It is important to note that the set of onsite interviews was first conducted on the experts' study group. Therefore some of the feedback produced by the first group had been already addressed by the time the interview was conducted on the second study group.

5.1.1 First Study Group: Expert Feedback

The results obtained while interviewing experts in human-computer interaction are qualitative, describing overall feedback and changes proposed for each page view.

The problems pointed out by the experts for each view and the solutions applied for each problem are described below:

- The scrolling within the Build tool makes it hard to drag the objects from the dataset pool to the boxes.
  Solution: As a solution for this problem, the placement of the dataset pool and boxes was changed to eliminate the need for scrolling, such that their views are side-by-side.

- The title of the current dataset that the user has chosen should be displayed in each Build and Explore view to provide better navigation through the website.
  Solution: The name of the current dataset was added.

- Descriptions should be added for each comparison method in Build view to make the purpose of each comparison method more clear.
  Solution: The help button, containing the descriptions of each comparison method was added.

- Use clicking movement to place objects into the boxes, instead of dragging.
  Solution: The suggestion was not implemented due to its conflicting nature with the logic used for Build tool. Even though the clicking action allows for fast object displacement, it does not provide a possibility to order objects manually within the box for List Comparison, etc.

- Scroll bars should be made visible every time you have a scrollable object.
  Solution: Scroll is visible at all times for all the scrollable boxes for Windows OS users and visible when hovered over for Mac OS users.

- The landing page does not convey enough information about the purpose of RANKIT.
  Solution: Landing page redesigned to show the instructions with icons on how to use the tool, and the descriptions of RANKIT's purpose.

- The results produced in Explore tool need to be explained to the user.
  Solution: The help button including instructions is added to the Explore view, the attribute weights are displayed as charts and shown in a separate tab.

- The user should have an ability to go back to Build tool to see how the rankings affected Explore results.
  Solution: The browser history is preserved, such that the user can come back to Build and Explore with his/her input. The URL saves the user's input.

- Non-numerical attributes should be included into the datasets to produce the results that would make more sense to the user.
  Solution: Categorical attributes are included into some datasets. The machine learning algorithm uses numerical and non-numerical attributes to produce the results.

- Take into account people with color-blindness.
  Solution: an extra box around buttons was added to make it less greyed out. The color scheme is picked to accommodate color-blindness.

5.1.2 Second Study Group: Usability

The results obtained while interviewing the second study group are quantitative. The subjects in this group consist of college students. The detailed results of the study can be found in the Appendix section A.

According to results obtained and shown on Figure 22, people used List Comparison method the most when first being exposed to all methods and later asked to choose the most prefered one among them. One possible reason for this distribution could be due to the fact that during the interview users are asked to test the List Comparison method first. This could have influenced some users to choose the List Comparison method in Task 2 of the onsite interview. In order to

address this possible bias, the online survey was designed to ask a user to evaluate only one method at a time. The online survey participants are randomly divided into three groups. Each group is given only one ranking method.

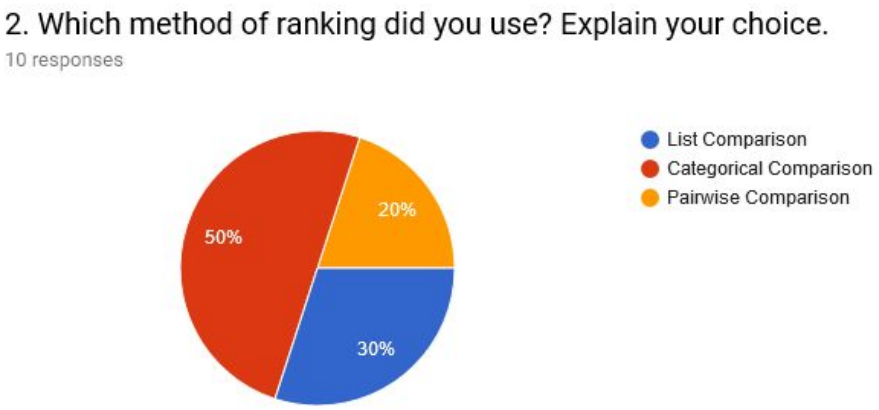
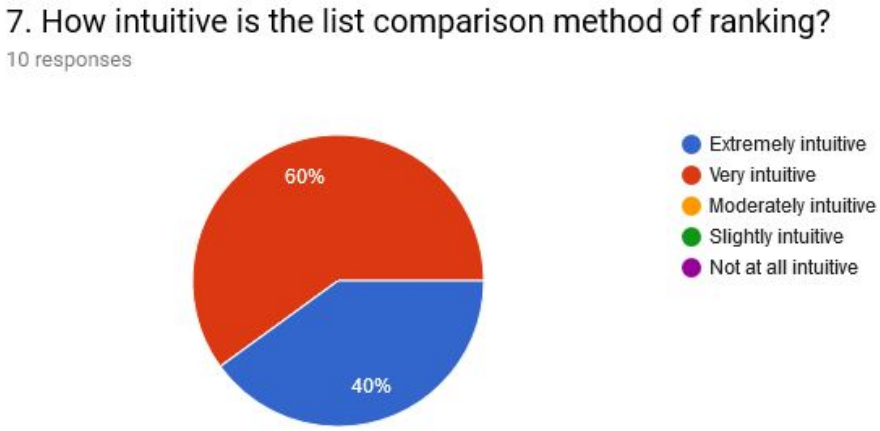


Figure 23. Most Widely Used Comparison Method

However, when asked about the intuitiveness of each method, the majority of the participants rated the Categorical Comparison method to be the most intuitive among three as shown on Figure 23. In fact, these results confirm the results obtained from initial informal user study showing the list and categorical comparison methods to be more widely used among participants.



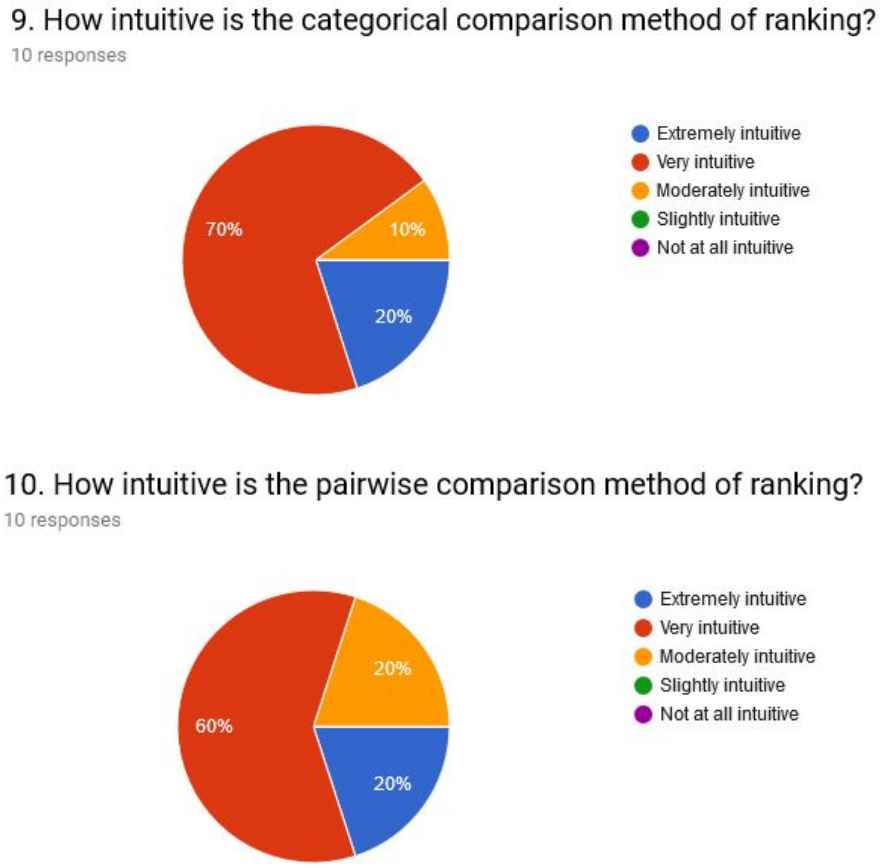


Figure 24. Intuitiveness of Comparison Methods

On average, the user ranks from 2 to 8 objects in general, and is willing to rank from 9 to 16 objects to get better results. Further analysis on the effectiveness of the overall ranking given a partial ranking of 9 to 16 objects should be conducted. It is important to note that the results for these two questions were directly affected by the level of participant's interest in the chosen dataset as shown on figure below.

Figure 25. Total Amount of Ranked Objects

Hence, if a participant was not interested in the given dataset, he or she would be ranking and willing to rank smaller amount of objects. Additionally, prior to conducting the study, the team made an assumption that the results for this section will indirectly show which out of three Build methods influences users the most to rank more objects. It was revealed that while the list and categorical comparison methods exhibited approximately the same influence on the user, the pairwise comparison method showed the least amount of influence on participant's willingness to rank more objects for better results out of three Build methods.

Most participants (70%) rated the purpose and function of RANKIT to be moderately clear, while 20% of participants gave the rating of the purpose to be very clear.

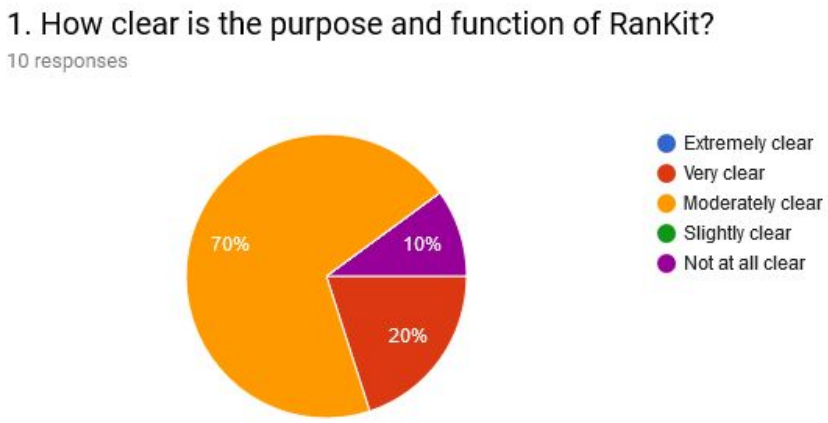


Figure 26. The Clarity of RANKIT's Purpose

When asked the reason of purpose being moderately clear, the majority of interviewed noted that it was due to the lack of attention they had when reviewing the landing page. However, after returning to the landing page view and giving a second look, they confirmed the purpose to be clear.

The features, such as search, scrolling within the dataset pool, and shuffle were reported to be working well and rated as not challenging to understand. It is important to note that while 60% of the individuals being interviewed found the understanding that drag and drop action should be used to rank to be not difficult at all, 30% of participants considered it as slightly difficult and 10% being moderately difficult as shown on figure below.

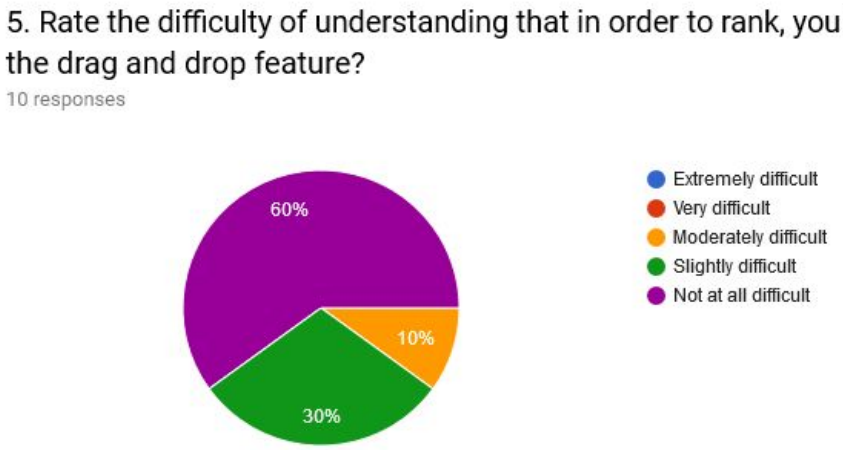


Figure 27. Drag and Drop Results

Based on these results, the conclusion can be drawn that for the user the drag and drop feature succeeded in being an intuitive way to rank objects. Some of the participants mentioned that clicking on the objects rather than dragging them could be used in order to decrease the amount of time spent on dragging objects from one box to another. However, this issue was not addressed as a future improvement since it contradicts the main purpose of users performing an interactive rank. In addition, clicking action does not allow for sorting within the ranking box for List Comparison method, as well as assigning objects to multiple boxes for Categorical Comparison method.

The “Colleges” dataset was revealed to be the most popular dataset among participants. The final question of the user study revealed that 10 out of 10 of participants would use the tool again. Additionally, the participant commented that they would be interested to use RANKIT as a personalized recommender system. Users had positive feedback on the tool’s flexibility because they have an option of multiple datasets and method for ranking to choose from.

5.2 Online Survey

At the end of the online survey, we have collected data from a total of 73 students who completed the online survey. The survey randomly assigned one third of the respondents to work with the List Comparison method, another third to Categorical Comparison method, and the rest to Pairwise Comparison method.

5.2.1 Evaluating the Intuitiveness of Comparison Methods

According to the results, over 60% of the respondents found the Build tool to be intuitive. When looking at the data more closely, 70% of the respondents who said that the Build tool is “extremely intuitive” were assigned the List Comparison method. When looking at the individuals who responded that the Build tool was “very intuitive,” the majority used either the List Comparison or the Categorical Comparison method. However, looking at the individuals who said that the Build tool was “Not at all intuitive”, it was revealed that they used the Pairwise Comparison method. Since the majority of individuals who mentioned that the tool was not intuitive used the Pairwise Comparison, it can be concluded that this method of ranking is more difficult for users to understand. This might lead them to not prefer to use this method of ranking.

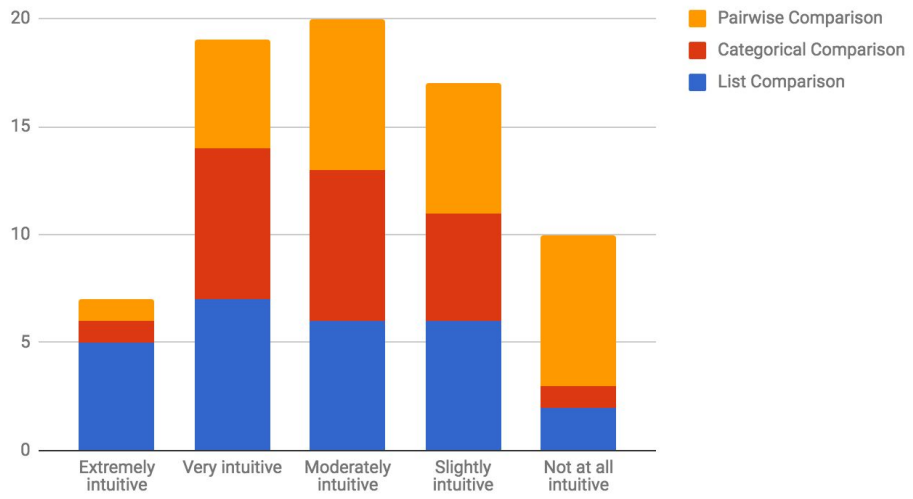


Figure 28. Intuitiveness of Comparison Methods

When analyzing the amount of objects users input into the ranking tool, it can be seen that a significantly large number of respondents who had to use the Pairwise Comparison method input less than 8 items. These results confirm the assumption we made based on the results from informal user study that the Pairwise Comparison method is not a common way people think of ranking.

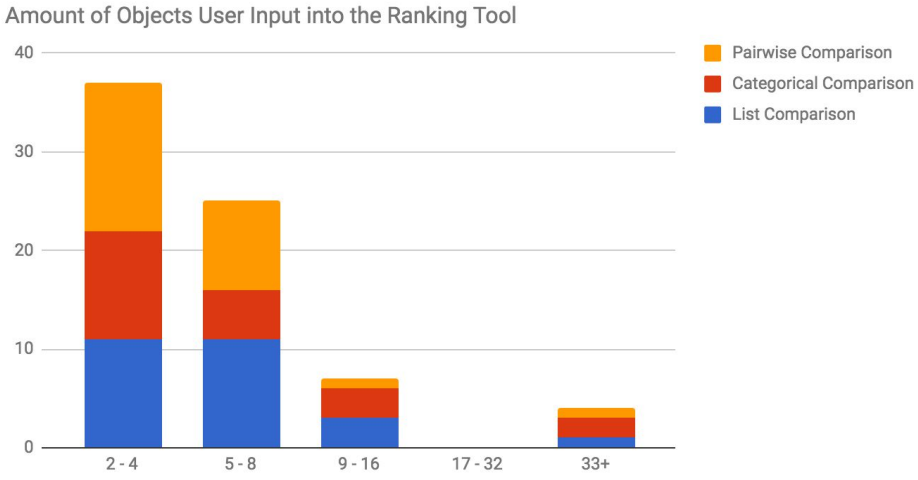


Figure 29. Amount of Objects Users Input into the Ranking Tool

These responses indicate that despite its complete control over comparing objects, Pairwise Comparison method is the least effective method of ranking. This is because users do not believe it is intuitive and do not wish to input more objects than they would for other comparison methods even though it is a method that needs the most items to be compared.

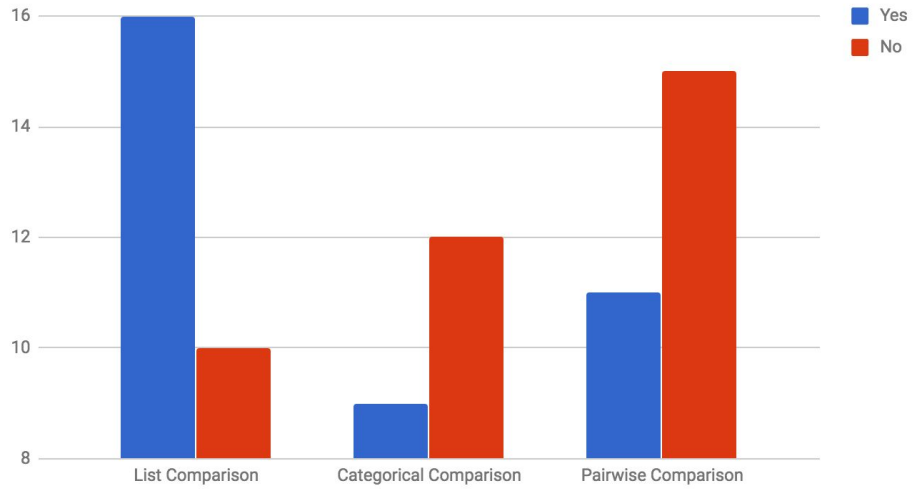


Figure 30. People Who would use RANKIT in the Future

Additionally, over 60% who used List Comparison responded they would use the tool again, while only 40% who used Categorical and Pairwise Comparison methods responded they would use the tool again. Therefore, the team concluded that people find the List Comparison method the most intuitive and the Pairwise Comparison method is least prefered.

5.2.2 Estimating the Amount of Information from Users

To estimate the amount of information users are willing to give to the machine learning system, the amount of objects users actually input is compared to the amount of objects users are willing to give to get better results. For List Comparison, 85% of users compared fewer than 8 objects while 75% of them are willing to input upto 16 objects in order to get better results.

Figure 31. Actual Input and Willing to Input: List Comparison

For Categorical Comparison, 75% of users compared fewer than 8 objects while 80% of them are willing to input up to 16 objects in order to get better results.

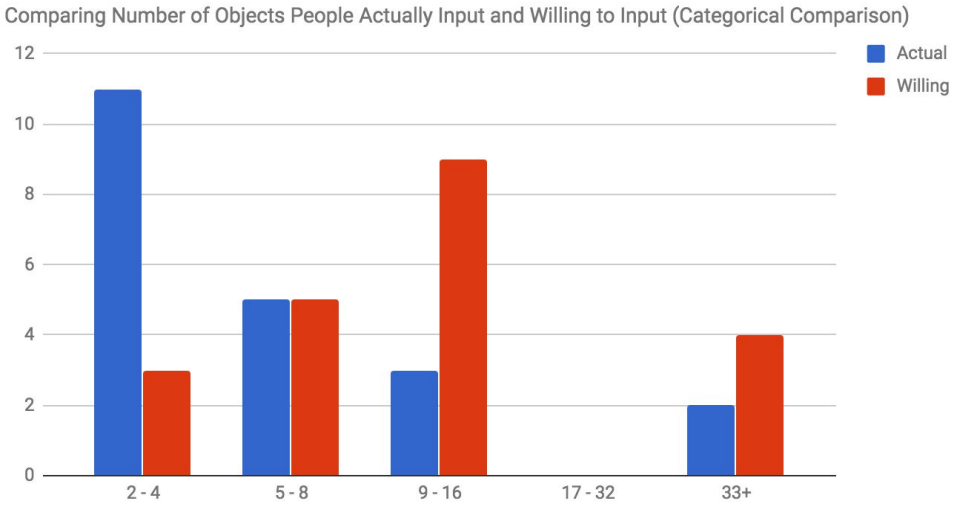


Figure 32. Actual Input and Willing to Input: Categorical Comparison

For Pairwise Comparison, 92% of users compared fewer than 8 objects while 90% of them are willing to input up to 16 objects in order to get better results. Since the drag and drop feature is more prominent in Pairwise Comparison and previous studies revealed the difficulties of drag and drop, the drag and drop feature could be discouraging users from using Pairwise Comparison and ranking more than like they would for either List Comparison or Categorical Comparison.

Figure 33. Actual Input and Willing to Input: Pairwise Comparison

According to the results, there is not a significant difference in number of objects users input across different comparison methods. In aggregate, 80% of the respondents inputs fewer than 8 items and 50% inputs fewer than 4 items. In order to get better results, 85% of the respondents are willing to input fewer than 16 items across all three comparison methods.

Additionally, the responses show that the amount of data users are willing to input is just one item number bracket higher than they initially inputted. Thus, it can be concluded that users are not willing to input much more than they initially chose to. These results can also indicate the flaws in the online survey, since the way these two questions are asked can create a bias in users, causing a bias in the results. Even though 16 items might be enough to give accurate rankings for small-to-medium-sized datasets, it is not enough to achieve decent rankings for large datasets with hundreds of objects.

5.2.3 Assessing Usability of RANKIT

According to the results, over 90% of respondents are satisfied with search functionality. Despite the clear positive response from people, only half of respondents figured out to use drag and drop to compare items. The other half, especially mobile users, either did not figure out or did not find the drag and drop intuitive. Additionally, only half of the respondents would use the shuffle feature to inspire their choices over items while the other half find shuffle feature not useful. Therefore, we concluded that majority of non mobile users finds the user interface of RANKIT usable and intuitive. The reason why the responses from mobile users are not taken into account is due to the fact that RANKIT was developed solely for use on laptops and computers. In addition, in the online survey it was explicitly stated not to use RANKIT on mobile devices.

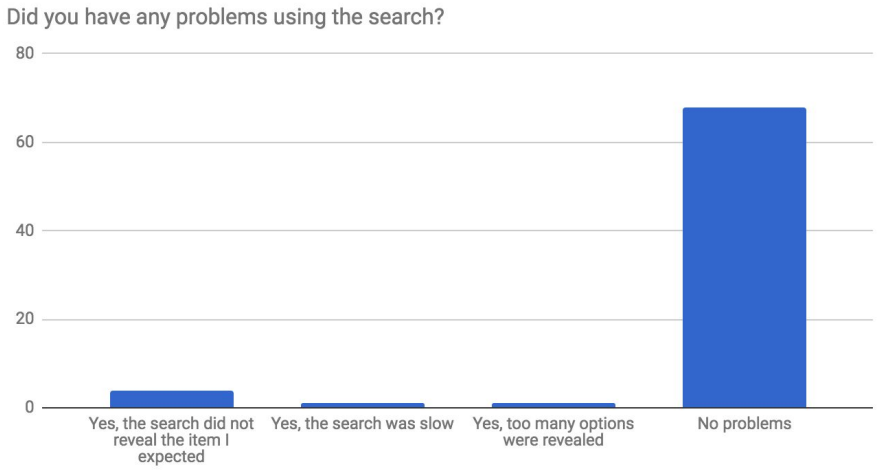


Figure 34. Responses for Search Feature in the Build Tool

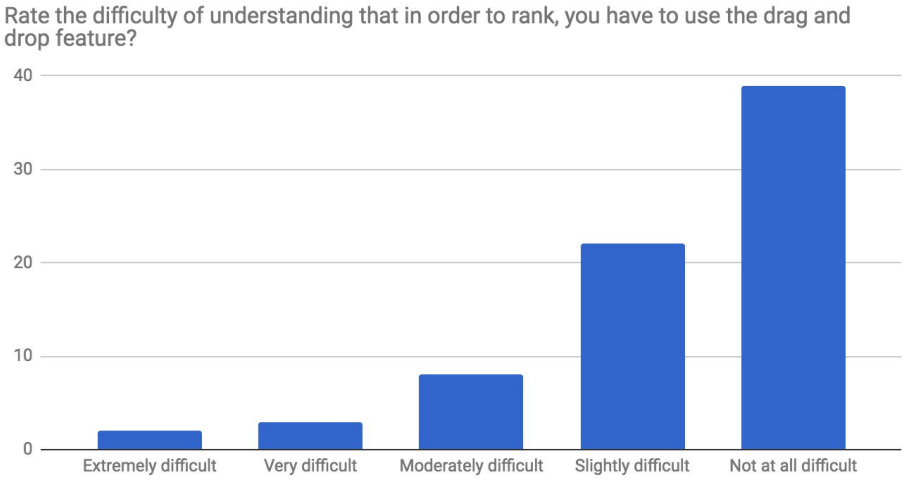


Figure 35. Responses for Drag and Drop Behavior in the Build Tool

Respondents also gave some feedback once they completed the online survey. Main criticisms came with the usefulness of the shuffle button in the build component of RANKIT. Users were mostly confused on why they would need to use the feature not knowing the team's intent for the tool was to help removed bias data selection when viewing a small section of the data set. While the functionality was fine and served its purpose, the need for shuffle came across as mostly unnecessary for many users. Ideally, looking into improving the usefulness of the Shuffle feature for audiences would be interesting to pursue in future work.

Another point users mentioned was the difficult nature of the website usability for mobile devices in that some of the features are not responsively designed. For example, one user claimed that using drag and drop on mobile devices was extremely cumbersome and overwhelming at times when ranking data. While it specifically mentioned in the survey to not use mobile devices when providing feedback (as the team was aware of the tool’s difficult usability), users do bring up good points in that almost a large audience for many websites consists of mobile phones and so building an interface that doesn't work in this way could disenfranchise a majority of the studies target audience. Therefore, redesigning the web application with responsive design to assist in sizing as well as alternative options to drag and drop data would make for an important future works upon future releases with the tool.

6. Conclusion

6.1 Summary

In the age of technology, the amount of data being exchanged amongst people are exploding exponentially. As a result, people are overwhelmed by overloaded information and facing difficulty finding signal among noises. To find the data and rank them according to the user's relevance becomes a crucial part of everyday life. Yet, rankings released from institutions and media are not useful to individual users, partly due to over generalization of interests, and partly due to the complex nature of ranking data itself.

Therefore, our researcher focuses on ranking tools which are transparent, personalized and relevant to each individual user. Our goals for this research were to design and evaluate an intuitive way of building partial rankings, as well as to estimate the amount of partial information users are willing to give.

To conduct this research, the team built RANKIT, a web application with python backend. Users provide a particular form of partial rankings and the system extrapolates the complete ranking using machine learning.

After the prototype, we asked UI/UX experts to critique RANKIT. After evaluating the feedback, we conducted onsite interviews with 10 users and an online survey with 74 respondents. The majority of online survey respondents found List Comparison to be the most intuitive ranking method, and the Categorical Comparison method the next most intuitive. Pairwise Comparison was not received as favorably, and users did not find it intuitie or easy to use. Both user studies show that majority of people ranked fewer than 8 data entries and are willing to rank up to 16 data entries. Even though 16 data entries is a small amount of data to train on for the machine learning system, perhaps future work can design improvements to allow the tool to request more data and train the model over time to achieve better results.

6.2 Future Work

In this section, the team address the areas where future researchers can research and improve RANKIT in terms of efficiency, usability and features.

Visualization of Explore

RANKIT currently displays the results of rankings as a simple data table. Weights of each attribute as a bar chart. The user experience would be greatly increased if the explore view is integrated with more interactive tools such as LineUp, where users can interactively change the weights of each attributes and reflect the result immediately.

Machine Learning Script

Future research on improving the speed and accuracy of ranking script can be carried out to further user adoption. As the user study revealed, individuals prefer to give the minimal amount of training data, but still expect a great level of personalization and accuracy. Therefore, techniques for determining the same level of accuracy for less amount of rankable data should be researched.

While the machine learning method utilized Pairwise Comparisons on objects as a whole, a different method of learning can be implemented. For starters, some individuals might not be familiar enough with whole objects or they might want to avoid biases they would have for those specific objects. To resolve such a problem, ranking by an attribute and ranking by multiple attributes can be explored. In ranking by an attribute, users will see the values of an attribute for all objects, ranking those values or assigning the same ranking for similar values. In ranking by multiple attributes the users will be able to rank objects that have the values of two or more attributes exposed.

Datasets

RANKIT currently has 6 datasets with different backgrounds, specifically Board Games, College in United State of America, FIFA Soccer Players, Movies, States in US, and Video Games. In the future, more datasets from reliable sources could be added to meet a variety of users' interests.

In order for RANKIT to be more open and flexible, uploading users' own data either from the storage (such as hard drive or cloud) or from a particular open data sites such as Kaggle could be implemented ("Datasets" [APA], n.d.). In this way, users have control not only over how data entries are being ranked, but also over which data entries are being ranked.

6.3 Team Experience

The team consists of four undergraduate students (Goutham Deva, Diana Doherty Malika Nurbekova and Zarni Phyo), two mentors (Professor Elke Rundensteiner and Caitlin Kuhlman) and an advisor (MaryAnn VanValkenburg).

Since all of the team members have technical background related to computer science, the team did not have any serious problem building a relatively complex system as RANKIT. Without knowledgeable observations from our two mentors, insightful guidelines from our advisor, coordinated team management from Diana and diligent input from each of the teammates, the team would not have been able to complete this research project. This project provided research in understanding human activity by measuring user's willingness to rank and concluding that actively, users are not willing to provide much information even if it means they will get better results.

7. References

Board Games Dataset. (n.d.). Retrieved from https://www.kaggle.com/mrpantherson/board-game-data

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005, August). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning* (pp. 89-96). ACM.

Chapter 20: Choosing an Application Type. (2009). Retrieved from https://msdn.microsoft.com/en-us/library/ee658104.aspx

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

FIFA Dataset. (n.d.). Retrieved from https://www.kaggle.com/artimous/complete-fifa-2017-player-dataset-global

Gehrke, D., Turban, E. (1999). Determinants of successful website design: relative importance and recommendations for effectiveness. *Systems Sciences. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference*.

Gnatyk, R. (2017). Desktop or web application | Exoft Software Development Company. Retrieved from http://exoft.net/desktop-or-web-application-what-to-develop

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).

Herbrich, R., Graepel, T., & Obermayer, K. (1999). Support vector learning for ordinal regression.

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

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.

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

Li, H. (2011). A short introduction to learning to rank. *IEICE Trans. Inf. Syst.*, 94, 1854–1862.

Liu, T.-Y. (2009). Learning to Rank for Information Retrieval, *Foundations and Trends Information Retrieval,* 3(3), 225-331.

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

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

Pahikkala, T., Tsivtsivadze, E., Airola, A., Boberg, J., & Salakoski, T. (2007). Learning to rank with pairwise regularized least-squares. In *SIGIR 2007 workshop on learning to rank for information retrieval* (Vol. 80, pp. 27-33).

States Dataset. (n.d.). Retrieved from http://davis.wpi.edu/dsrg/PROJECTS/MATTERS/index.html#

Stone, D., Jarrett, C., Woodroffe, M., Minocha, S. (2005). User Interface Design and Evaluation, 1st edition.

Szummer, M., & Yilmaz, E. (2011, October). Semi-supervised learning to rank with preference regularization. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 269-278). ACM.

Video Games Dataset. (n.d.). Retrieved from https://www.kaggle.com/rush4ratio/video-game-sales-with-ratings

Wall, E., Das, S., Chawla, R., Kalidindi, B., Brown, E.T., Endert, A. (2013). Podium: Ranking data using mixed-initiative visual analytics. *IEEE transactions on visualization and computer graphics*, 24(1), 288-297.

Why to Use Node.js: Pros and Cons of Choosing Node.js for Back-end Development. (2017). Retrieved from https://www.netguru.co/blog/pros-cons-use-node.js-backend

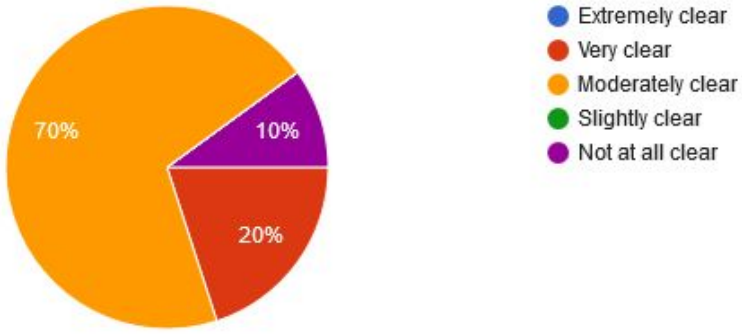
Zehlike, M., Bonchi, F., Castillo, C., Hajian, S., Megahed, M., & Baeza-Yates, R. (2017, November). Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (pp. 1569-1578). ACM.

Appendix

Part 1: Familiarizing with the Build Tool Components

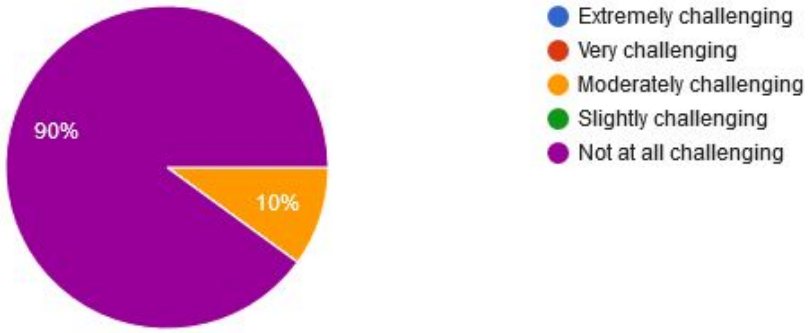
1. How clear is the purpose and function of RanKit?

10 responses



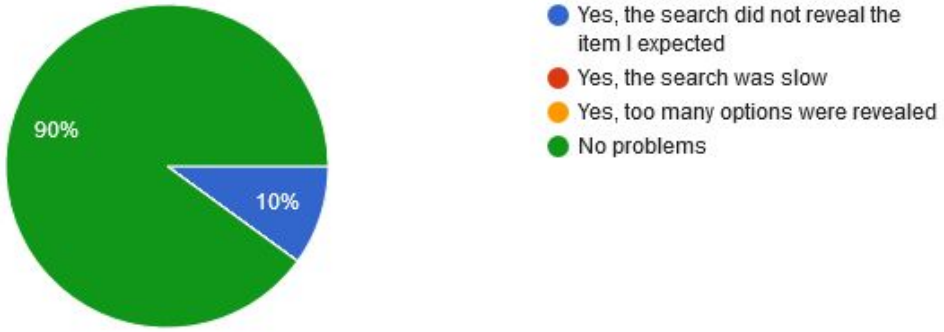
2. How challenging was it to understand that more movies could be revealed by scrolling?

10 responses



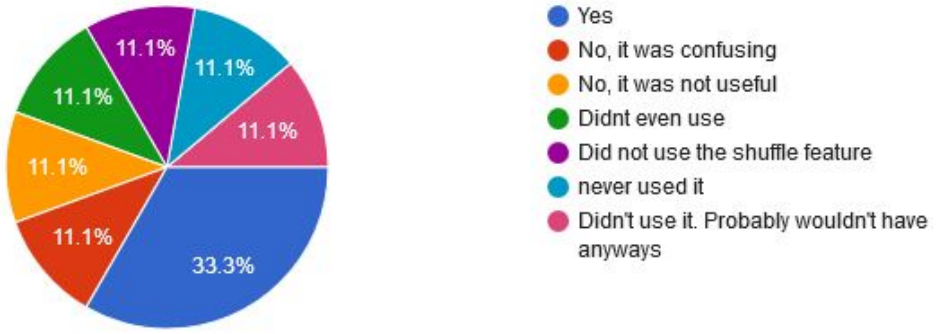
3. Did you have any problems using the search?

10 responses



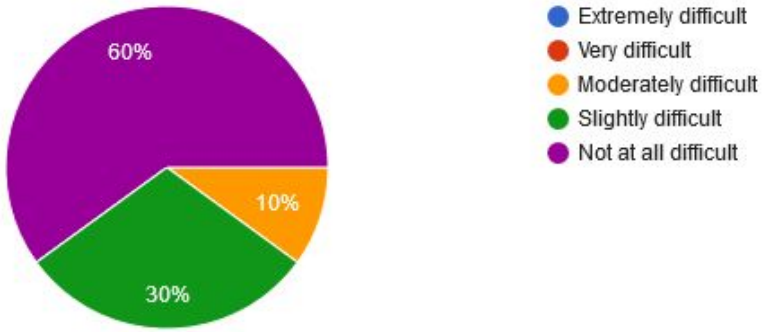
4. Would you use the shuffle feature again to inspire your future choices?

9 responses



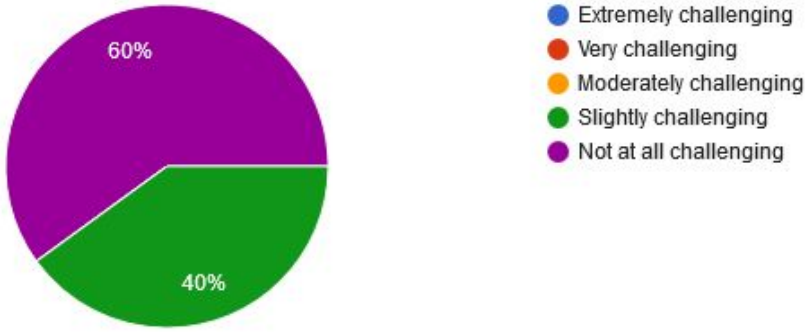
5. Rate the difficulty of understanding that in order to rank, you have to use the drag and drop feature?

10 responses



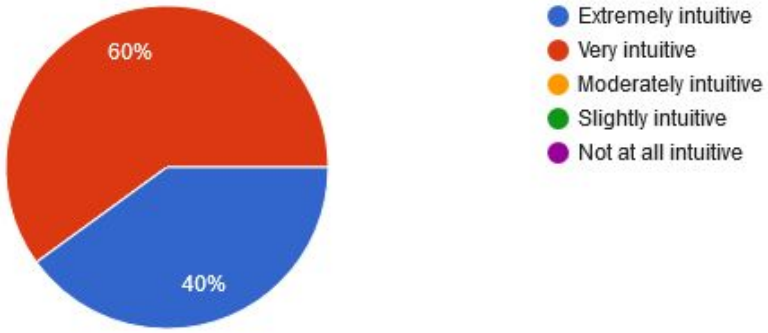
6. How challenging was it to use the drag and drop feature?

10 responses



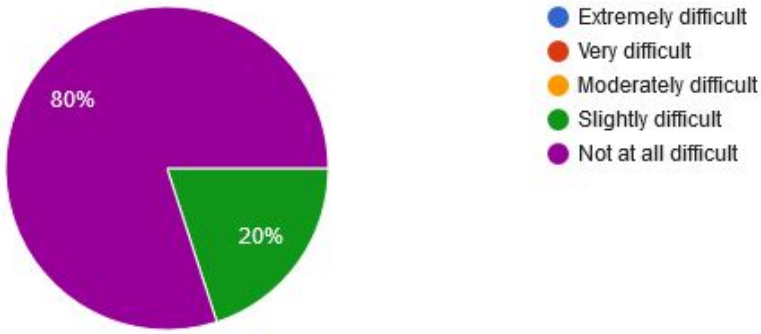
7. How intuitive is the list comparison method of ranking?

10 responses



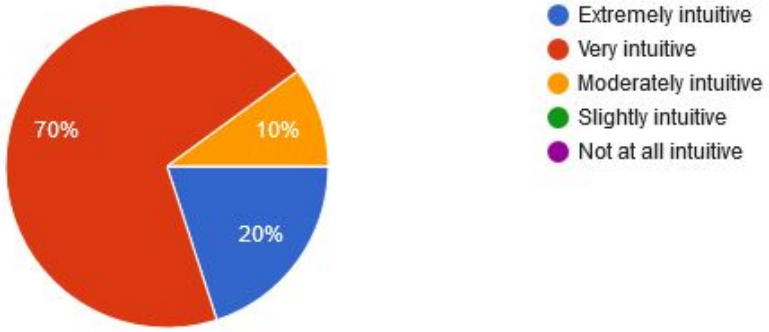
8. How difficult was the transition from one method to another?

10 responses



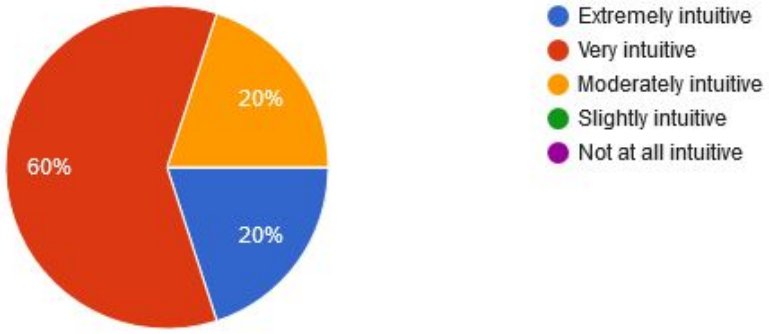
9. How intuitive is the categorical comparison method of ranking?

10 responses



10. How intuitive is the pairwise comparison method of ranking?

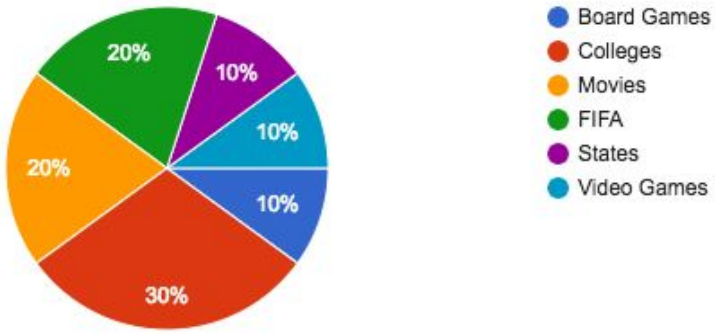
10 responses



Part 2: Exploring a Specific Build Tool Component

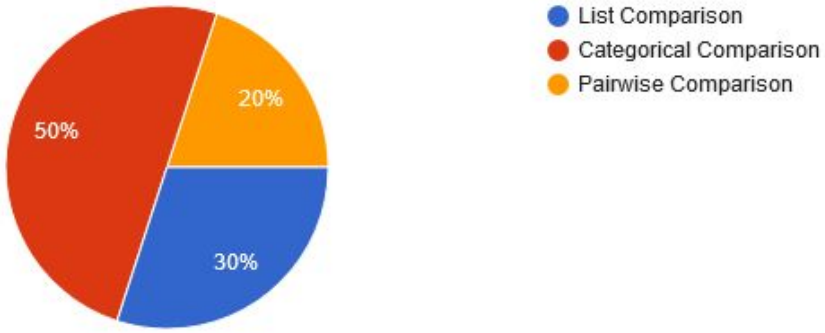
1. Which Dataset did you use?

10 responses



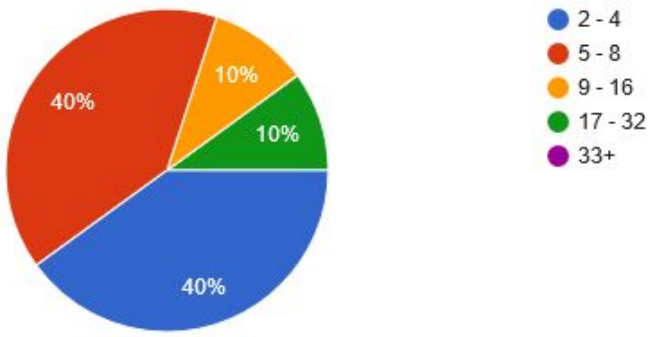
2. Which method of ranking did you use? Explain your choice.

10 responses



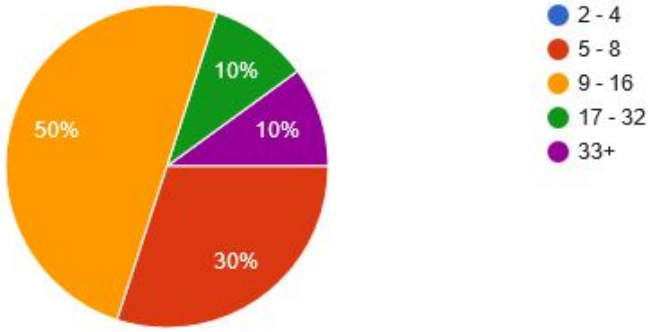
3. How many objects did you input into the ranking tool when you were given the option to rank a familiar dataset?

10 responses



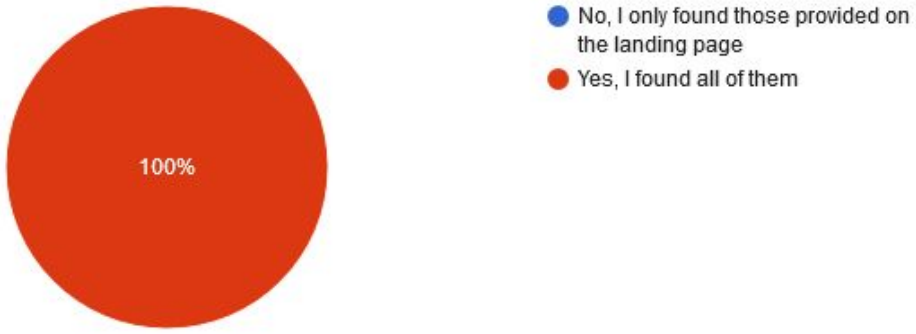
4. How many objects are you willing to rank in order to get better results?

10 responses



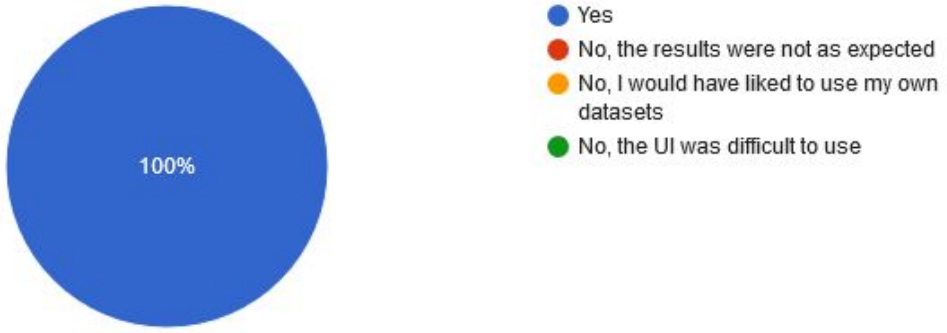
5. Were you able to find all the available datasets?

10 responses

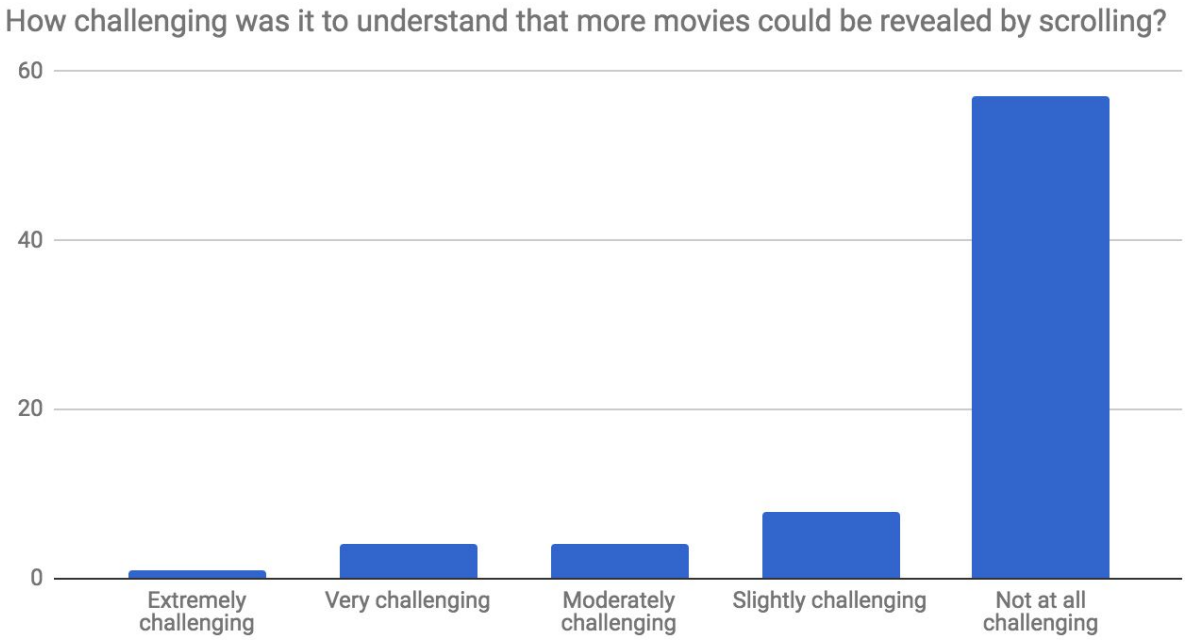


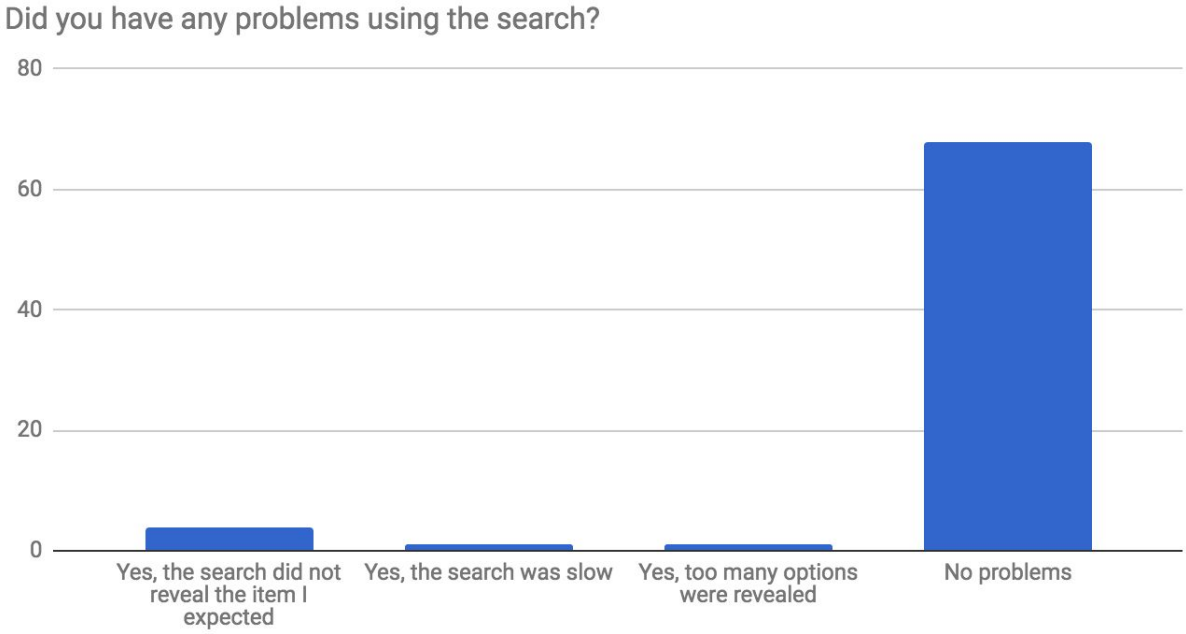
6. Would you use this tool again? Optional: Elaborate more on why or why not.

10 responses

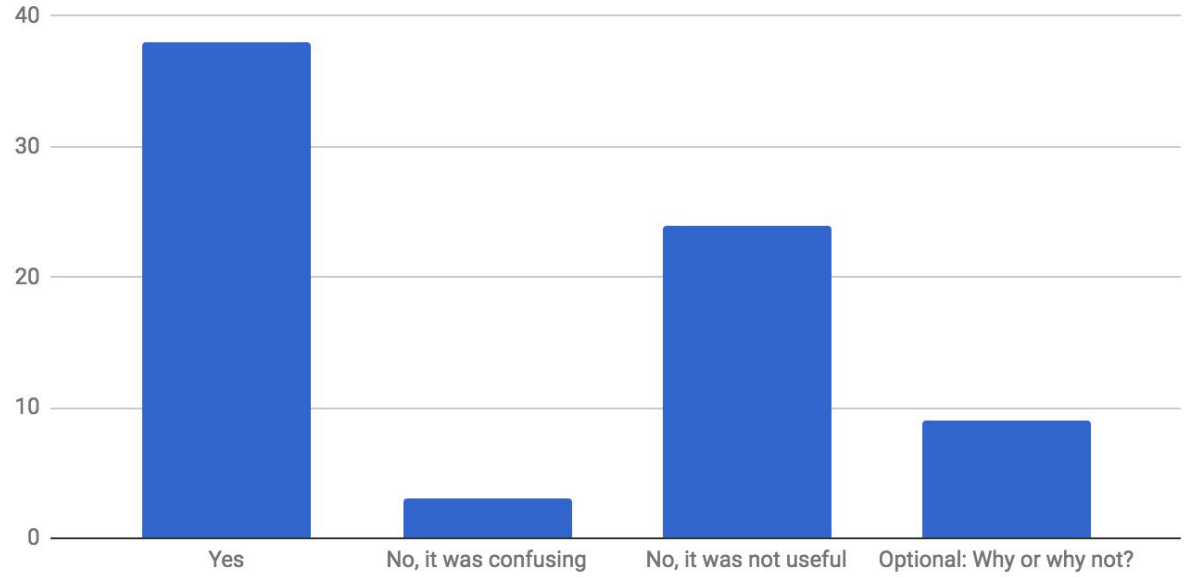


Online Survey

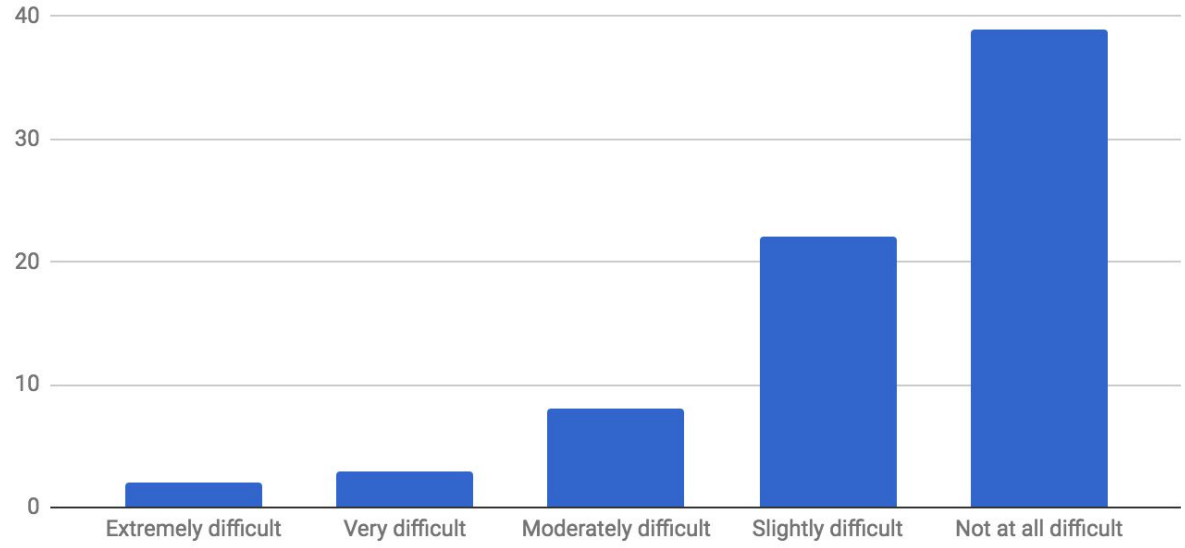




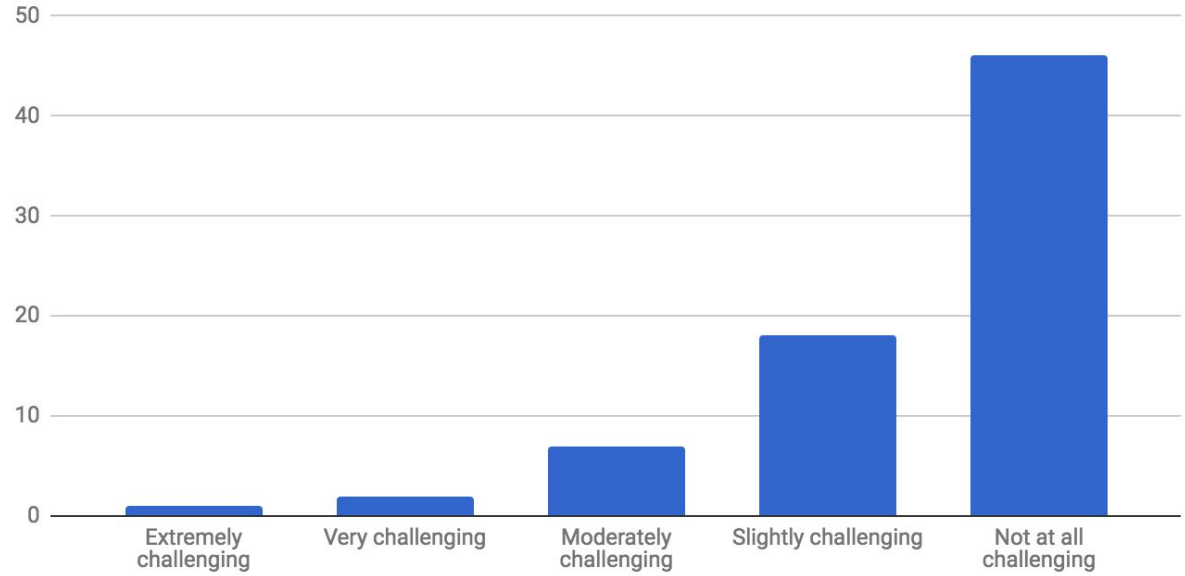
Would you use the shuffle feature again to inspire your future choices?



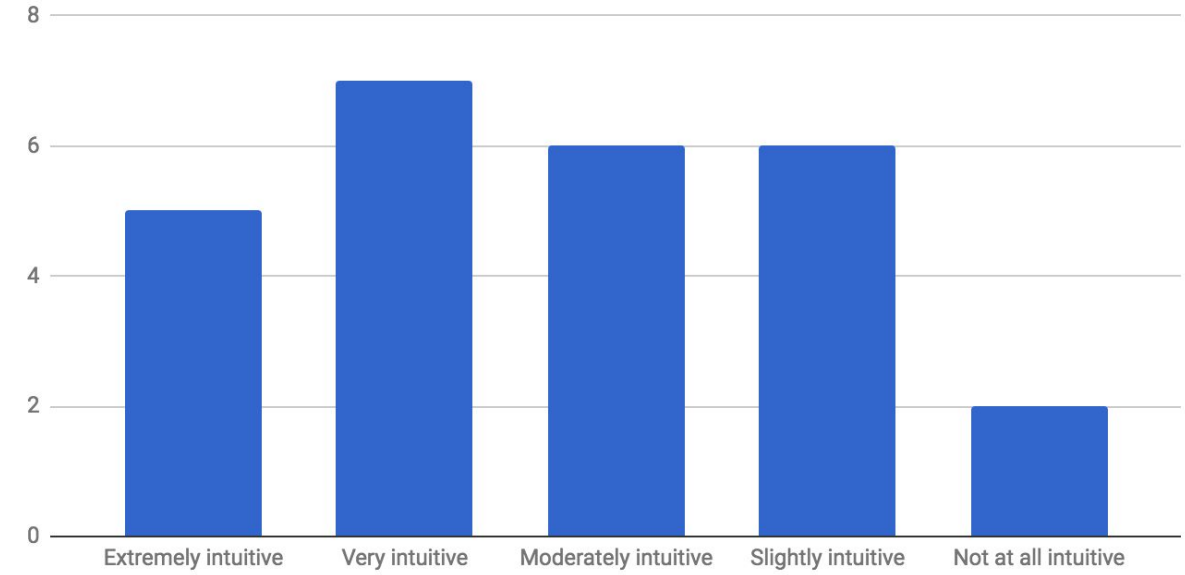
Rate the difficulty of understanding that in order to rank, you have to use the drag and drop feature?



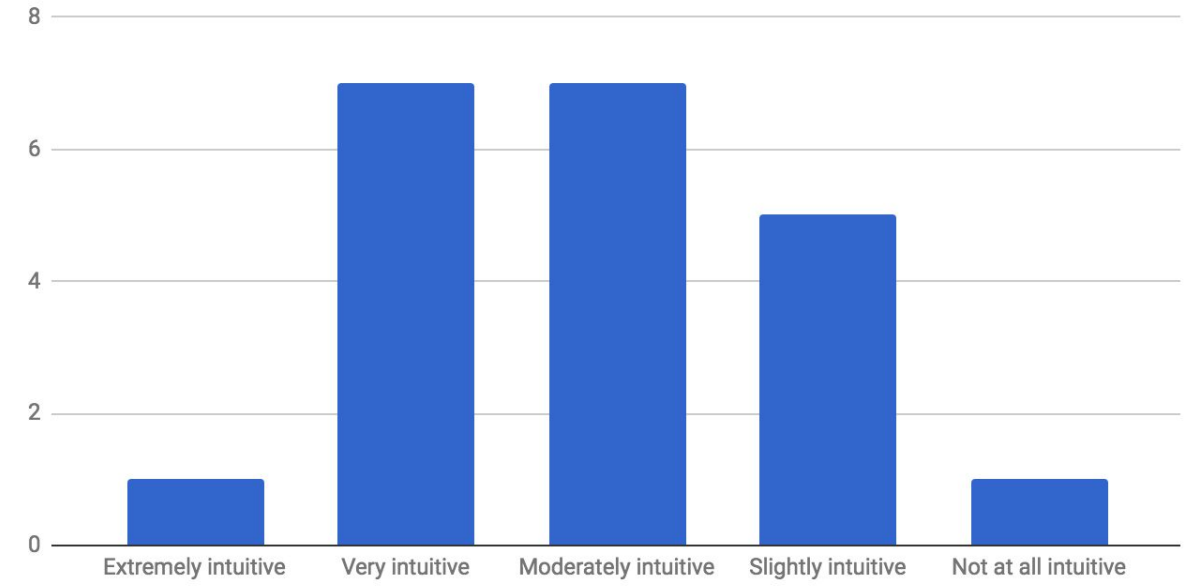
How challenging was it to use the drag and drop feature?



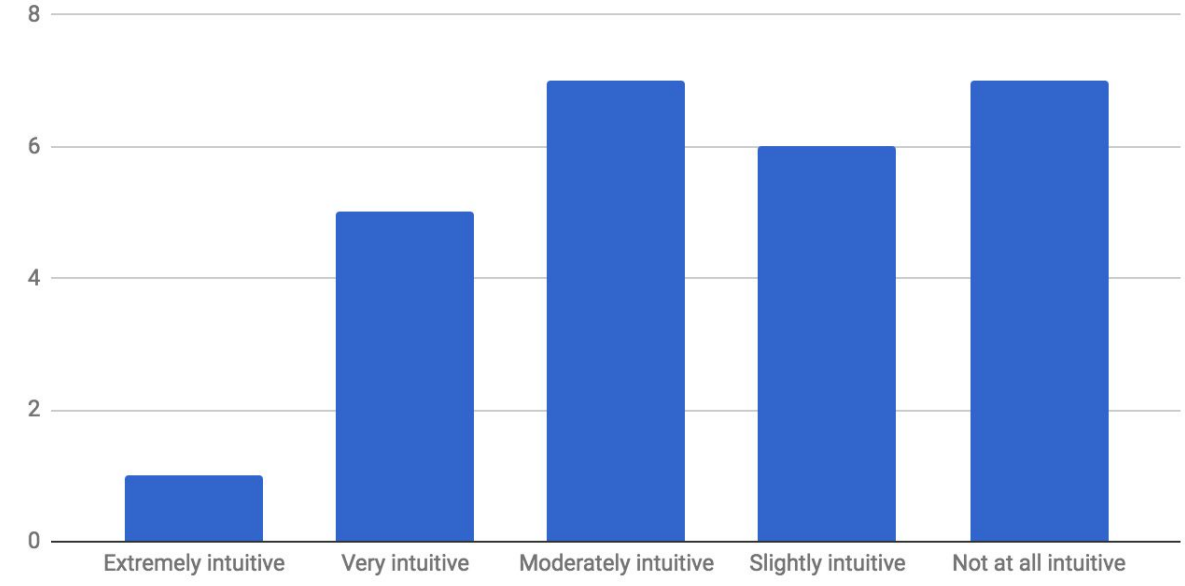
How intuitive is List Comparison method of ranking?



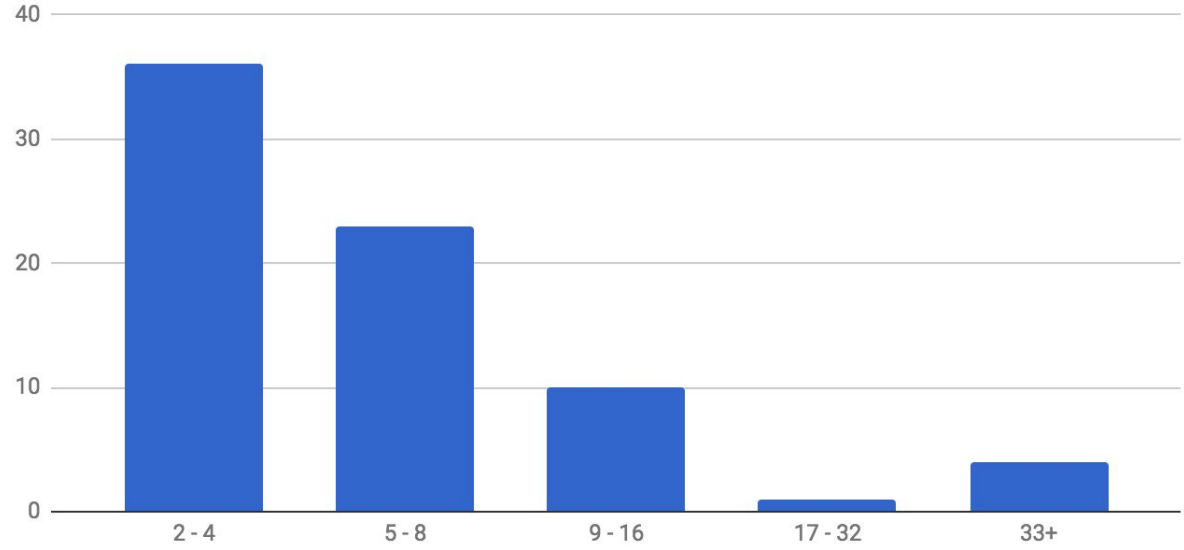
How intuitive is Categorical Comparison method of ranking?



How intuitive is Pairwise Comparison method of ranking?



How many objects did you input into the ranking tool when you were given the option to rank a familiar dataset?



How many objects are you willing to rank in order to get better results?