

Podium: Ranking Data Using Mixed- Initiative Visual Analytics

Bharath Kalidindi, Emily Wall, Subhajit Das, Ravish
Chawla, Eli T. Brown, and Alex Endert

Abstract

Ranking points of data is utilized in everyday decision making, and multi-attribute ranking systems are a tool used to facilitate the ranking process and help make these data-driven decisions. These systems ask users to assign weights to the attributes for representing the value each attribute to a decision, which the system then uses to compute a ranking of the data. However, it is not always easy or even possible for users to quantify or understand the relative importance of each attribute to the dataset. In fact, people generally have a more holistic understanding of the data. To address these challenges, we present a visual analytic application to help people rank multi-variate data points. We developed a prototype system, Podium, that allows users to drag data points in a table to positions within the ranking they assign based on their perception of the data points value, and generate a model based on their initial ranking that represents their perception of the data. We use Ranking SVM to make these inferences and build this model that generates the attribute weights. We also present how our system can be used to understand user preferences as well as deconstruct existing rankings.

Introduction

Ranking is a widely studied process within the field of visual analytics. It can be a useful tool for users to better understand how data points relate to one another, and how the attributes of the data contribute to the ranking. For example, critics rank films based off of actors' reputations, cinematography, story themes, and other factors, placing various levels of importance on each of these considerations. Current visualization techniques, such as LineUp [4] or ValueCharts [2], allow for the user to adjust the attribute weights to create a ranking. This provides the user an intuitive approach to create an ordering for a dataset that could be large in terms of size and space and tedious to do manually. These techniques are particularly effective when the user understands beforehand which attributes are influencing their projections of the data points rather than understanding how various data points relate to each other.

Most people, however, are better at determining how various data points relate to each other rather than the magnitude at which the attributes contribute to their decisions. Carterette et al. demonstrated that people are cognitively skilled at making relative judgments on data items [3]. Because users generally struggle to pinpoint how weights reflect their own decision making in terms of ranking individual data points, the resultant user set weights often will not reflect their personal preferences. Users may also identify various points to upgrade or downgrade in the ranking, and can only do so by precisely tweaking the attributes. Creating a system which allows

the user to move these data points, creates a model for inferring attribute weights, and creates a general ranking based on the resultant weights would be more beneficial in this scenario.

Our research aims to address this problem of utilizing the data point relations users understand intuitively to create rankings that reflect how the user perceives the dataset to better understand their own preferences and biases in relation to the dataset attributes. We will present a novel prototype called Podium, a multi-attribute ranking system which allows the user to interact with data points, uses a model within the domain of learning to rank algorithms to generate a new attribute weight vector, and apply the newly derived weights to the dataset to create a new ranking representing the user's perception of the dataset.

Literature Review

Our visualization system touches on several domains that need to be addressed to create Podium. There have been several examples of multi-attribute systems in the past as well as systems employing mixed-initiative analytics such as the one we will be providing for the user to interact with the data. We are also using a machine learning algorithm to extract a new attribute weight vector once the user interacts with the data points. The visualization and the backend work together to generate to the user their preferences based on their decisions.

Multi-Attribute Ranking Systems

Options for visualizing multi-attribute rankings are already available. These visualizations help users identify the most important data points from a large dataset as well as aid the user's decision-making based on the attribute values and the attribute weights. Visualization techniques such as LineUp [4], TableLens [10], and ValueCharts [2] provide such multi-attribute ranking interfaces for the user. Table Lens uses graphical representations for a table that supports a very large amount of data in a single view. The user is able to interact with the table to focus on areas of importance. Table Lens utilizes a fisheye technique to provide this view by dynamically altering the layout of the view to focus on areas of higher importance without loss of information and maintaining label information. LineUp and ValueCharts support data rankings based on multiple heterogeneous attributes with adjustable scales. Users of

visualizations like these can combine attributes and set parameters to explore how various combinations and configurations result in various rankings. The method in which the user creates a new ranking of the dataset tends to be by changing the attribute weight and columns manually, which may not be as intuitive for the user to find a ranking he or she agrees with.

When the user surveys the results of the ranking, he or she may identify a data point misplaced, and may want to move it higher or lower within the dataset. However, current systems do not have the functionality to generalize attribute weights based on the user's interaction with the dataset. The user is generally limited to interacting with the columns or attribute weight. Podium is built around allowing users the option to rank a subset of the dataset they are familiar with, and use the underlying factors of this ranking to create a generalized ranking for the entire dataset.

Mixed Initiative Visual Analytics

Recently, there has been development in mixed-initiative systems that can infer user intention from their interactions with the dataset. Based on the principles presented by Horvitz [5], mixed-initiative visual analytics focuses on balancing human and machine effort while performing a visual data exploration task. Purely automating these tasks can result in erroneous results and lack of user trust, while complete lack of automation results in a heavy workload for the user. Dis-function [1] is an example of such a system, leveraging automation to learn distance functions. Users directly interact with points on a scatterplot in order to correct the output. The system creates models based off these interactions to better capture the user's underlying intentions. InterAxis [7] and AxiSketcher [8] similarly employ mixed-initiative principles. These systems utilize simple interactions, such as click-and-drag, to define parameters for custom dimension reduction on a scatterplot. In general, these examples of mixed-initiative systems leverage a small sample of user interactions, approximate an analytical model, and use that generalized model to organize and visualize the remaining data.

Our system employs mixed-initiative visual analytics to provide users a model of their preferences. The user will first drag data points to new positions within the table holding the ranking, and the system will compute a new attribute weight vector based on these interactions. The user won't be required to rank every data point in the system, as that could be overwhelming depending on the size of the dataset. These guidelines are in accordance with the balance

between human and machine effort mixed-initiative visual analytics strives for, while also showing how mixed-initiative approaches can be applied in a multi-attribute ranking system to model user preferences.

Machine Learning

Our system will use a machine-learning algorithm to retrieve the attribute weight vector. Learning to rank using machine learning is generally approached through three techniques: pointwise, pairwise, or listwise [9]. Pointwise approaches find a ranking by creating scores for individual data points using regression and classification algorithms. Liu found that pointwise approaches performed consistently worse than pairwise and listwise approaches for ranking benchmark document retrieval data [9]. Pairwise approaches use pairs of data points and an indication of the ordering within the pairs to train a model and create a ranking aiming to maintain the pairwise ordering. Listwise approaches look at the entire dataset to obtain an optimal ordering which will minimize a loss function defined by the nature of the ranking. While Liu also advocated for listwise approaches given a full dataset [9], our system will only be handling a subset of the dataset, making a pairwise approach more suitable.

In our case, the model is trained using a pair of data points along with a corresponding label to indicate which data point was ranked higher. Input data transformed in such a way makes it suitable for a machine learning algorithm such as Ranking SVM, which has been used previously for better understanding the underlying forces behind a ranking [6]. An algorithm such as this would provide a set of attribute weights that could be applied to the entire dataset. Because a pairwise approach is being utilized, the final ranking will attempt to maintain the relative ordering for the points interacted with by the user.

Notation	Description
$\mathbf{a} = \{a_1, \dots, a_m\}$	set of m attributes describing D
$\mathbf{w} = [w_1, \dots, w_m]$	attribute weight vector; $\mathbf{w} \in \mathbb{R}^m$
$D = \{\mathbf{d}_1, \dots, \mathbf{d}_n\}$	dataset of size n ; $\mathbf{d}_i \in \mathbb{R}^m$
$r(\mathbf{d}_i)$	rank score of data point \mathbf{d}_i
k	number of rows used to train the Ranking SVM model
$\hat{C}(d_{i,j})$	normalized contribution of attribute a_j to the rank score of data point \mathbf{d}_i
\hat{x}	normalized value of x

Table 1: Notation used to describe the ranking solver

Methods and Algorithms

Models

We derive an attribute weight vector based on the user’s interactions with the data using a Ranking SVM model [8]. This weight

vector represents how the users interprets each attribute value of data points, exposing the users preferences towards the attributes used to rank data points. A positive weight indicates higher attributes values are preferred, while a negative weight indicates lower values are preferred. A Simple Additive Weighting (SAW) model is used to create the final ranking of the data points based on the weight vector.

Ranking SVM utilizes the process of support vector machine (SVM) for the ranking problem. The implementation of SVM found in the python library Scikit-Learn is used within our technique. SVM sets the data points to some m dimensional space with corresponding labels represented as a set of tuples, (d_i, y_i) for data point $d_i \in R_m$ and label $y_i \in \{-1, 1\}$ for a two-class problem. The output model is a hyperplane, defined by a vector in R_m , that cuts through the space of the data. The model is optimized so that points with $y_i = -1$ are on one side of the hyperplane, points with $y_i = 1$ are on the other, creating a wide margin of separation between points and the plane.

Ranking SVM applies the idea of optimizing for a hyperplane to the ranking problem with pairwise constraints. While Standard SVM utilizes a full set of labeled data points, we use a subset of data point pairs d_i and d_j , and a label telling us if d_i is considered better than d_j . Ranking SVM will be inputted the difference vectors for pairs of data points, $d_i - d_j$, and it will attempt to predict which point is better based on their difference. Equation 1 describes how the data is inputted given the pair (d_i, d_j) .

$$\left(\mathbf{d}_i - \mathbf{d}_j, \begin{matrix} 1 \text{ if } \mathbf{d}_i \text{ is preferred,} \\ -1 \text{ otherwise} \end{matrix} \right) \quad (1)$$

The derived model using Rank SVM can take in a pair of points and predict which should be ranked higher. Although all of the constraints resulting from the user's interactions might not be satisfiable, it would be frustrating to not be able to provide any model for the user. This is why we have chosen to model all constraints as soft constraints rather than hard constraints, guaranteeing a set of attribute weights modeling the user's constraints as closely as possible by penalizing constraint violations.

Because our model learns from the users' interactions, it may simply provide confirmation of their initial biases of the data. However, the primary purpose of the system is to allow an exploration of the data with previous biases exist and gain a better understanding of the world in which their perception is truth. For example, a user may arrange data points in the table

to create a model where a particularly favored data point is the top ranked item. When the system attempts to model the user's preferences, the favored data point may remain on top, but the attribute weights do not fit the user's initial perception, giving the user new insight into the data point and perhaps into why he or she favors it.

Ranking

Using Ranking SVM's model we can now rank the data points within our dataset. The difference vectors of two data points are passed to the Ranking SVM classifier (e.g., for points i and j , $(d_i - d_j)$) to predict the ranking order of the two input points. The dot product of the given difference vector with its internal model, a weight vector w , is used to determine the classification of the point. If the dot product $w \cdot (d_i - d_j)$ is positive, the difference vector belongs to the positive class $y = 1$, and therefore d_i belongs relatively before d_j in the ranking. If the dot product is negative, the difference vector belongs to the negative class $y = -1$, and d_j belongs relatively before d_i in the ranking. Pairwise combinations of vectors are passed into Ranking SVM in this way until a full ranking of the data is produced.

The Ranking SVM model can be used in this way to generate a ranking, using its output on any pair of data points as a comparator function to sort in $O(n \log n)$. However, rather than obtaining the relative rankings, we gain flexibility by calculating a score for each individual point based on the Ranking SVM model and ranking based on scores. Equation 2 shows how we use the dot product to generate individual rank scores.

$$r(\mathbf{d}_i) = \mathbf{w} \cdot \mathbf{d}_i = \sum_{j=1}^m w_j d_{ij} \quad (2)$$

These individual rank scores are then sorted, with the highest value corresponding to the top rank. The result is ultimately the same as if we had used Ranking SVM directly to define the data points' classes since $w \cdot (d_i - d_j) = w \cdot d_i - w \cdot d_j$. If $w \cdot d_i > w \cdot d_j$, then $w \cdot (d_i - d_j) > 0$ so d_i is ranked above d_j , and vice versa. Whether the weights come from the Ranking SVM model or through the user's adjustments through the interface, this alternative approach allows us the flexibility to use the same ranking algorithm for the points regardless of the origin of the weights.

Podium

Interface

The interface for Podium allows users to navigate the various functionalities of the application and understand the output with ease. The main table presents the dataset used within the system. Within the table, the rows represent the data points and the columns represent the attributes for the data points. The data are ordered by their ranking calculated as described in the methods and algorithms section. To re-position data points within the table, users can drag and drop a row to a new position. Left of the main table is the control panel, which contains visualization and model controls for creating new weights and rankings. The *Compute Weights* button derives a new set of attribute weights based on the user's interactions with the rows in the table. The *Rank* button re-ranks all the data points based on new rank scores derived by the attribute weights.

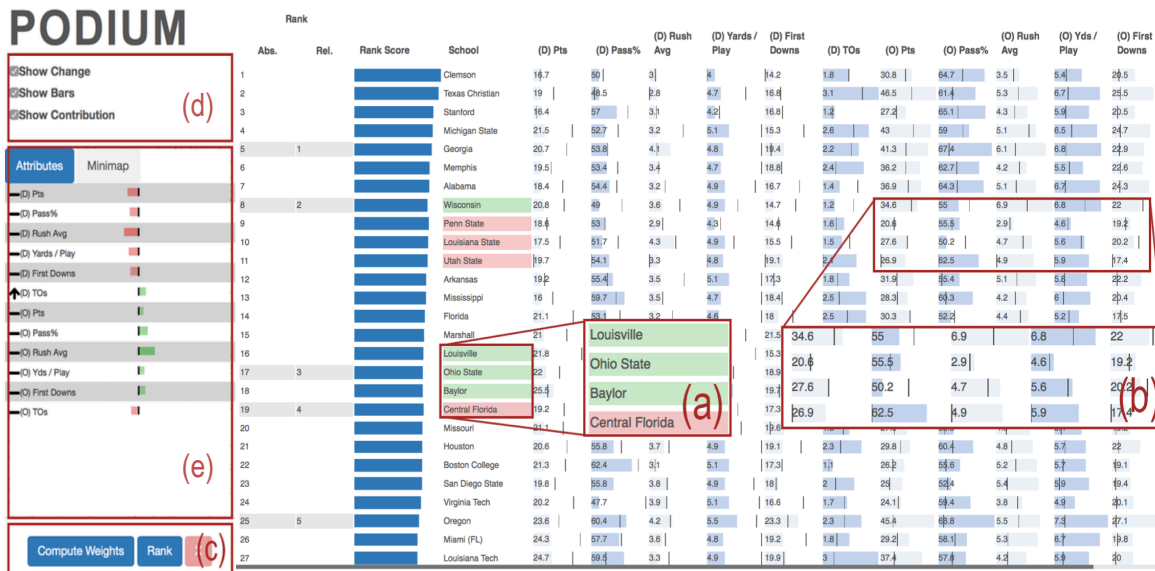


Figure 1 Overview of Podium interface

Figure 1 shows the main table and control panel. The visual encoding options Show Change, Show Bars, and Show Contribution are selected in the control panel (d). (a) When a user changes the position of a row (row 19), Show Change changes the colors of the rows with new positions, with red encodings for rows moved down and green encodings for rows positioned

higher (e.g., rows 16- 18). (b) Show Bars encodes attribute values as the width of the bars, and Show Contribution adds the thin vertical black bars that represent each attribute's contribution to the data point's rank score. (c) The model controls are *Compute Weights*, *Rank*, and *Discard* (represented as X). *Compute Weights* triggers the system to derive a new set of attribute weights based on the user's interactions with the data points in the table using the Ranking SVM algorithm. *Rank* generates a new ranking based on the current attribute weight vector, whether it's the result from *Compute Weights*, or manually modified by the user. *Discard* returns the system to the last saved state. (e) The attributes tab shows the attribute weight vector with corresponding bar representing its weight as a value between -1 and 1 (between -100% and 100%). The red bars represent negative weights and the green bars represent positive weights (these weights can be manually adjusted).

Usage Scenarios

Understanding Biases

Say we were attempting to understand why our favorite college football teams from 2014 were successful. Given a college football dataset that contains 128 different schools and contains 12 numerical attributes representing defensive and offensive statistics such as points, passing percentage, rushing average, yards per play, first downs, and turnovers, we can input our data into Podium and attempt to identify the most significant factors. Say we were Georgia and Florida State fans. We then move these teams up near the top of the table. We believe Auburn is a good team as well, but not quite at the level of Georgia and Florida State, so we position Auburn a bit lower. Georgia Tech and New Mexico State had poor seasons, so they are positioned further down. With our initial ranking of a subset of the teams, we press Compute Weights to derive an attribute weight vector based on the constraints for the teams. Figure 2 depicts the outputted attribute weights.

PODIUM

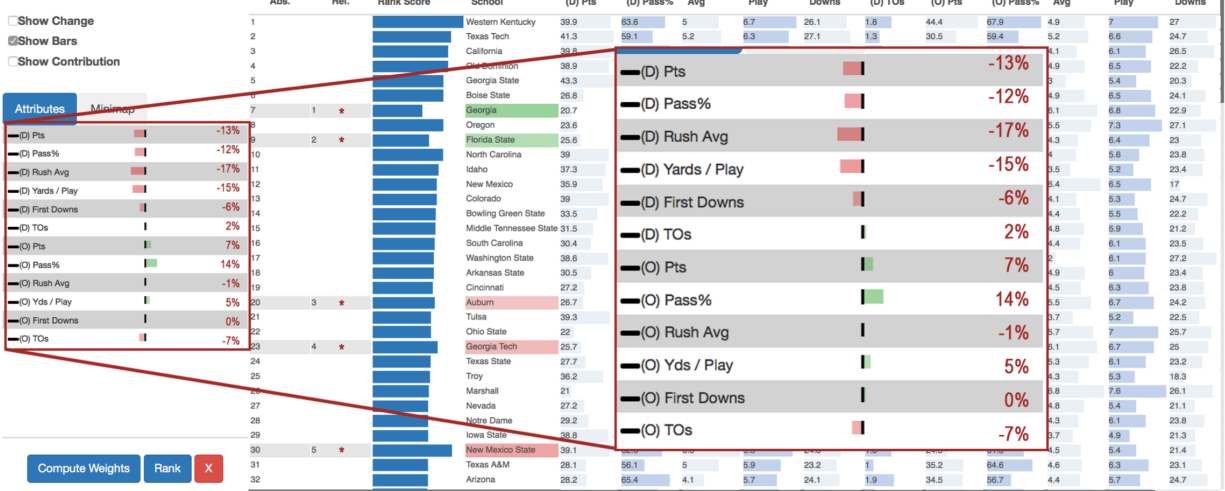


Figure 2 Initial interactions

We can then use our newly derived weight vector to generate a new ranking for our dataset by pressing Rank. The results are shown in Figure 3. The relative order of the teams is maintained in the ranking, with Georgia at #5, Florida State at #50, Auburn at #56, Georgia Tech at #96, and New Mexico State at #125.

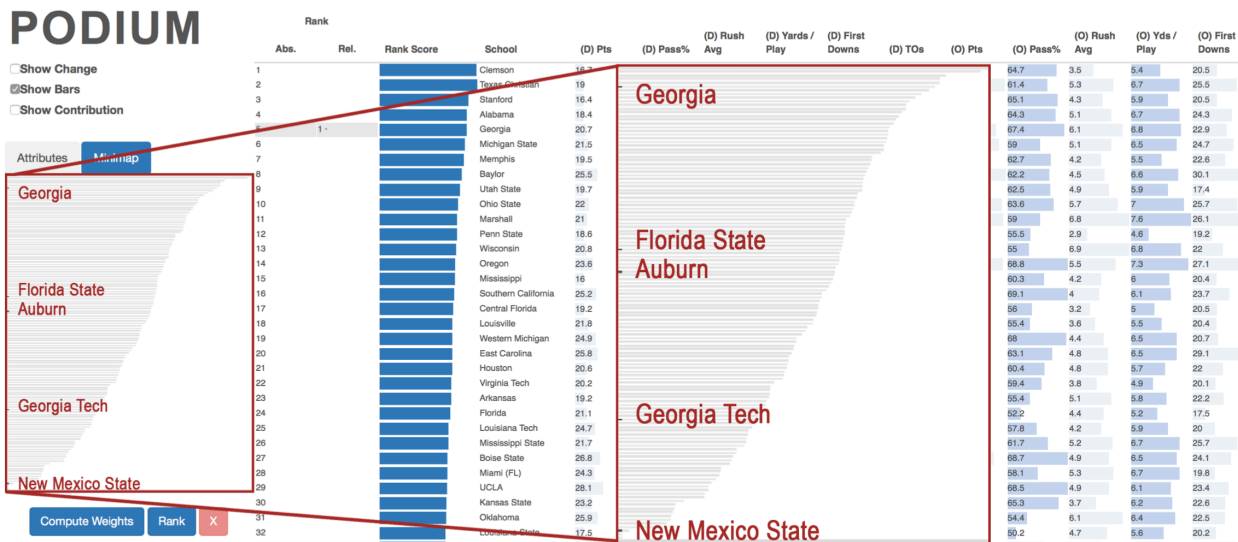


Figure 3 Changes to ranking based on inferred weights

Currently, Alabama is ranked higher than Georgia, but perhaps we wish to enforce it further to avoid Alabama slipping behind Georgia in the next derived ranking. We can click on the row to refine the model and reiterate Alabama's correct positioning to the ranking model.

Looking at our attribute weights, most of the offensive statistics are weighted positively except for rushing average. This is counterintuitive, as more rushing yards per play should be beneficial for a team. We toggle the arrow up (↑) to emphasize the attribute in the model, as depicted in Figure 4.

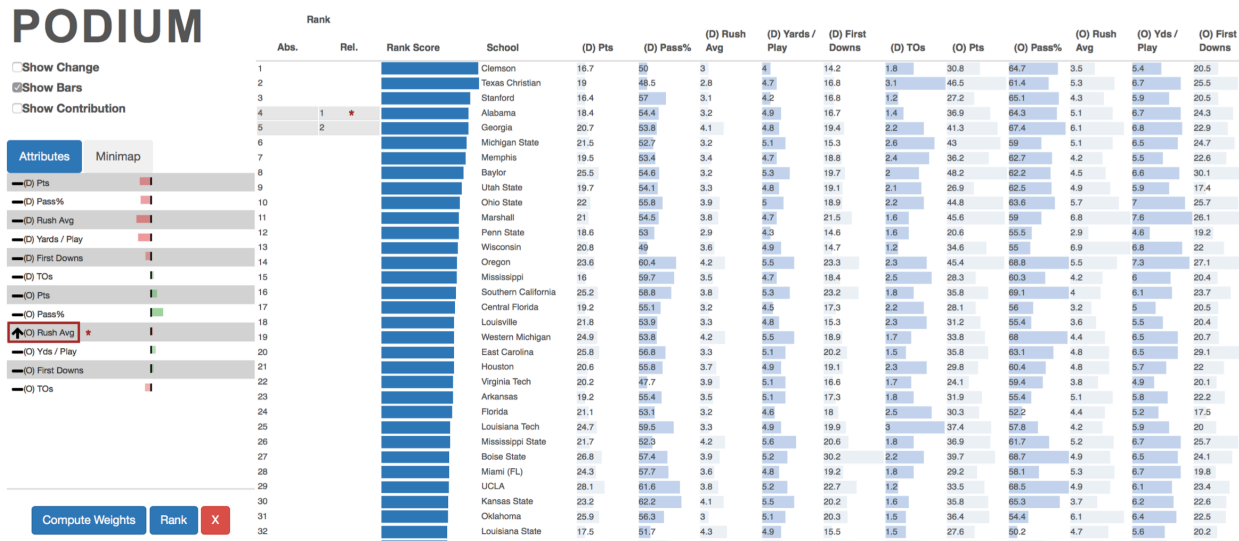


Figure 4 Change in ranking for Alabama and Georgia, with emphasis on positive value of rushing average

Once again, we press Compute Weights to regenerate the model using the new constraints and presses Rank to apply the resulting weight model to the full dataset. The final model increased offensive rushing average from -1% to 7% as well as slightly shifted some of the other attributes' weights. Since a negative attribute weight indicates that low values of an attribute are more valued, the shift to a positive weight for offensive rushing average is a good thing since higher values of rushing yards are preferred for good teams. Ultimately, the model resulted in Alabama at #4, Georgia at #5, Florida State at #49, Auburn at #58, Georgia Tech at #89, and New Mexico State at #125. If we enable the Show Contribution option in the control panel as shown in Figure 5, we see that while Alabama and Georgia possess strong offenses, their defense contributes far more to the success of her favorite teams.

PODIUM

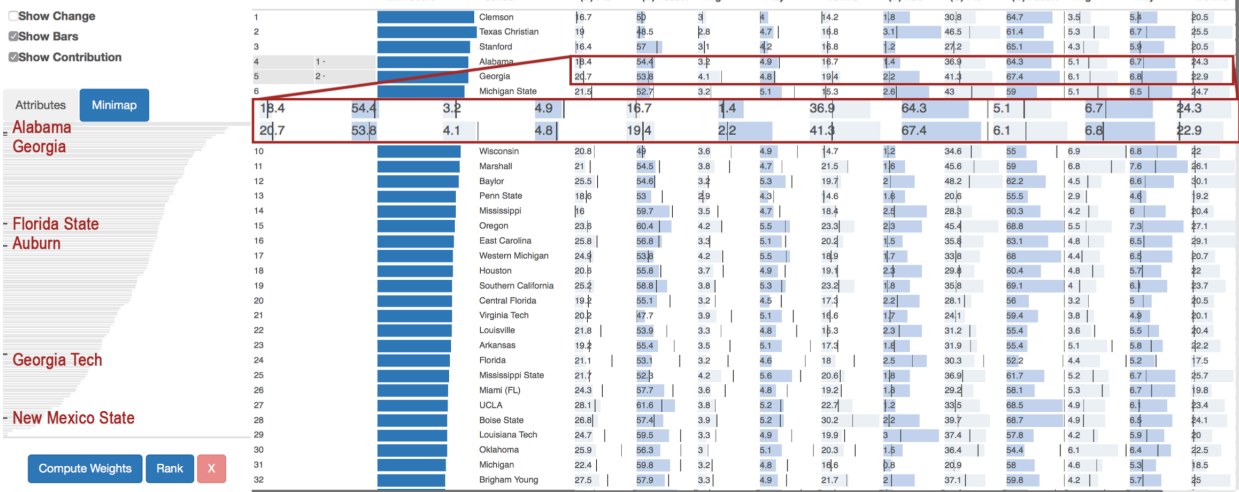


Figure 5 The final ranking

Through using Podium, we learned that our favorite teams were successful thanks to particularly strong defenses that allowed very few points, complete passes, and rushing yards. Offense was generally less important, with the exception of passing percentage at 14%.

Deconstructing Existing Rankings

We could also use Podium to understand the significant factors for successful teams in 2014 to make better predictions about successful teams in the future. Using the same 2014 college football dataset as in the previous usage scenario, we rank teams according to their final standings from the end of the season.

First, we look at teams in the Atlantic Coast Conference (ACC). Their ranking at the end of the 2014 season was, starting with the best, Florida State, Clemson, Georgia Tech, Louisville, Duke, Boston College, North Carolina, Pittsburgh, North Carolina State, Virginia Tech, Miami, Virginia, Syracuse, and Wake Forest. Figure 6 shows the results after dragging the teams to the correct position, and clicking Compute Weights, providing use the attribute weights to explain our ranking. Our weights show that winning ACC teams have strong offenses, with a large weight on offensive points and first downs, while defense wasn't quite as significant.



Figure 6 The final weights inferred using ACC standings

Preliminary Results

In order to analyze the effectiveness of our system, we looked for preliminary feedback on the attribute weight model as well as the general usability of our prototype. To obtain feedback, we conducted a session with 4 participants (P1-P4) who are experts in the field of information visualization. Each participant was given a quick tutorial on how to navigate the prototype, during which they were encouraged to ask questions as needed. They then took time to click around in the interface in order to familiarize themselves with the interactions and controls. Participants were then asked to perform a ranking task using a dataset of movies and reflect on the model results and interface presentation. Feedback came in the form of users expressing their thoughts out loud when interacting with the system.

Our system was built around a holistic approach, and this was found to be very useful for all the participants, as it prompted reflection of their own preferences and biases. P3 noted that the system was fun to use, allowing him to see how certain conceptual categories of movies he used for ranking resulted in very different attribute weight vectors. P3 also noted that the ability to derive attribute weights as well as manually tweak them was appreciated. P2 and P4 observed that Podium acted as an interactive recommender for movies in this task. By providing exemplary rankings, the generalization of their rankings to the full dataset brought movies to the top that they would like to see. P4 commented that this approach to ranking allowed her to not be

overwhelmed by specific attribute values of movies. All participants thought that the interaction technique of clicking and dragging rows in the table was easy to understand and found the Relative Rank column helpful for seeing how the model treated their constraints.

Participants tended to believe the model better reflected their subjective preferences as they interacted more with the system; however, they had different interpretations of the derived model when it disagreed with their mental model. For example, P1 doubted the accuracy of the underlying model, whereas P3 and P4 rationalized the model by referring to the data to make sense of the attribute weights. When the model resulted in a particular movie continually being brought back up to the top of the ranking, P2 stated it was like “having an argument with the model.” Despite this, P1 and P2 appreciated that the derived weight vector allowed them to see a quantitative representation of attributes that they valued. P4 reflected that the model-derived attribute weight vector made it clear she had a subconscious preference for newer movies. Additionally, participants took different approaches to handling movies that they were not familiar with. P2 ignored movies he had not previously seen, while P1 and P3 explicitly positioned movies lower in the ranking that they had not seen before. P1, P2, and P3 expressed some frustration about the movies they did not know about getting in the way of the movies they did want to rank. P4 commented that it would have been nice to support tied rankings (e.g., in the case where she did not really know which of two movies was better).

Discussions

Interface

Our visualization system allows users to leverage their holistic preferences for data points and obtain a model for their preferences at the attribute level. Based on the feedback, our prototype was insightful for our users. However, building trust in the model is important in order to encourage users’ reflection in such cases. P4 noted that the system illuminated a subconscious preference she had for newer releases; however, P1 questioned whether the model was working properly. For future work we would like to better foster this process of self-reflection by incorporating an approach to help users better understand how to interpret the ranking results.

The technique we chose to use to model the attribute weight vector (Ranking SVM) only models the relative positions of data points. However, this is not easily understandable when operating the system. Based on preliminary user feedback, one interface alternative might be to rank the training rows in a separate staging area that did not contain the full dataset. Alternatively, temporarily collapsing rows that the user has no opinion about or otherwise removing the gap between rows that train the model might be beneficial. Future work might include implementing a similar technique to make the user fully aware of the importance of the relative ranking.

Ranking Solver

Ranking SVM is an effective model for pairwise learning to rank [9]. We chose a pairwise approach for learning to rank as users may find it difficult to accurately position data points at an absolute position. However, other pairwise learning to rank approaches exist including RankNet [2] and RankBoost [5], each with different strengths and weaknesses. Further, other types of user input might be more appropriate in other contexts. For example, if users want to rank order a full dataset, listwise models might be more appropriate. Similarly, if users have a strong notion of absolute position of data points or are deconstructing an existing ranking, it may be more effective to use pointwise models like regression. As future work, we plan to further scrutinize the appropriateness of different models for determining the attribute weight vector in various contexts.

Our observation sessions showed that users tend to interact with Podium for short periods of time. This provides our SVM smaller amounts of data for training, resulting in underfitting compared to a user's "ground truth" of what the ranking of the data should be. Future work would involve identifying more constraints for each interaction to provide more data for our model.

Conclusions

Ranking data points is one of the primary ways in which we comprehend a dataset in order to make decisions about the data. Previous systems assume that users are able to quantify their conceptual understanding of how important particular attributes are to a decision, but this is not always easy or even possible for users to do. People often have a more holistic understanding of the data points as a whole rather than the attributes that weight them. In this paper, we developed a prototype system, Podium, that allows users to drag rows in the table to rank order data points based on their perception of the relative value of the data. Podium then infers a weighting model using Ranking SVM that satisfies the user's data preferences as closely as possible. We present two usage scenarios to describe some of the potential uses of our proposed technique, and conducted a usage session to obtain preliminary feedback, which for the most part, validated our method.

References

1. E. T. Brown, J. Liu, C. E. Brodley, and R. Chang. Dis-Function: Learning Distance Functions Interactively. *IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 83–92, 2012.
2. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.
3. G. Carenini and J. Lloyd. ValueCharts: Analyzing Linear Models Expressing Preferences and Evaluations. *Working Conference on Advanced Visual Interfaces (AVI)*, pages 150–157, 2004.
4. B. Carterette, P. N. Bennett, D. M. Chickering, and S. T. Dumais. "Here or There." Preference Judgements for Relevance. In *Advances in Information Retrieval*, volume 4956, pages 16–27. Springer Berlin Heidelberg, New York, 2008.
5. Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.
6. S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit. LineUp : Visual Analysis of Multi-Attribute Rankings. *IEEE Transactions on Visualization and Computer Graphics*, (August), 2013.
7. E. Horvitz. Principles of Mixed-Initiative User Interfaces. *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, (May):159–166, 1999.

8. T. Joachims. Optimizing Search Engines using Clickthrough Data. *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142, 2002.
9. H. Kim, J. Choo, H. Park, and A. Endert. InterAxis: Steering Scatterplot Axes via Observation-Level Interaction. *IEEE Transactions on Visualization and Computer Graphics*, 22(1): 131–140, 2016.
10. B. C. Kwon, H. Kim, E. Wall, J. Choo, H. Park, and A. Endert. AxiSketcher: Interactive Nonlinear Axis Mapping of Visualizations through User Drawings. *IEEE Transactions on Visualization and Computer Graphics*, 2626(c):1–1, 2016.
11. T. Liu. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2007.
12. R. Rao and S. Card. The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information. *Proceedings of the SIGCHI conference on Human . . .*, (April):318–322, 1994.