Podium: Ranking Data Using Mixed-Initiative Visual Analytics

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

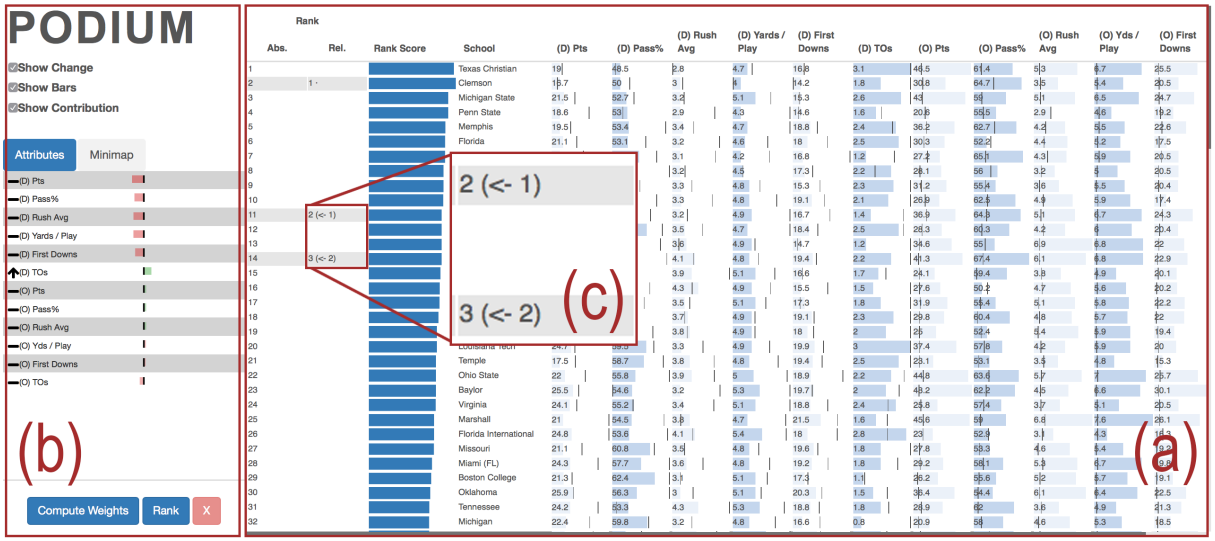


Fig. 1: The Podium interface contains two primary views: (a) the main table, which displays each data point as a row in the table and each attribute as a column; and (b) the control panel, which has controls for the visual encodings in the table, attribute weights, and the underlying Ranking SVM model. The callout box (c) shows the Relative Rank column, which displays the current relative position of the rows used to train the model as well as the previous user-defined relative rank. The interface is described in detail in Section 4.1.

Abstract—People often rank and order data points as a vital part of making decisions. Multi-attribute ranking systems are a common tool used to make these data-driven decisions. Such systems often take the form of a table-based visualization in which users assign weights to the attributes representing the quantifiable importance of each attribute to a decision, which the system then uses to compute a ranking of the data. However, these systems assume that users are able to quantify their conceptual understanding of how important particular attributes are to a decision. This is not always easy or even possible for users to do. Rather, people often have a more holistic understanding of the data. They form opinions that data point A is better than data point B but do not necessarily know which attributes are important. 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 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. Whereas past systems help users understand the relationships between data points based on changes to attribute weights, our approach helps users to understand the attributes that might inform their understanding of the data. We present two usage scenarios to describe some of the potential uses of our proposed technique: (1) understanding which attributes contribute to a user's subjective preferences for data, and (2) deconstructing attributes of importance for existing rankings.Our proposed approach makes powerful machine learning techniques more usable to those who may not have expertise in these areas.

Index Terms—Mixed-initiative visual analytics, multi-attribute ranking, user interaction.

1 Introduction

Ranking of data points is one of the fundamental analytic operations people perform as part of visual data analysis [1, 24]. Rankings are used for many reasons: to understand the most important items from a large dataset, to make a decision based on the attributes of the data, or to give meaning to data that otherwise has no inherent order. People rank sports teams based on past statistics or regional pride. Critics rank movies based on attributes like the quality of the cinematography, how captivating the story is, and the reputation of the director. Consumers rank cars based on horsepower, price, and fuel economy in order to determine the right purchasing decision. Cyber security analysts prioritize which network threats to analyze first. People also compare established rankings with their expectations for how good or bad individual data points are. For instance, experts publish rankings of college football teams based on performance in areas like passing yards, touchdowns, and penalties. Those rankings may bring up questions for fans, like why a particular team is at the top of a ranking given their low offensive statistics.

Ranking models allow people to order a large set of data points and gain an understanding of how data points relate to one another at the

- *Emily Wall, Subhajit Das, Ravish Chawla, Bharath Kalidindi, and Alex Endert are with Georgia Institute of Technology, Atlanta, GA, USA. E-mail: {emilywall, das, rchawla, bharkal, endert}@gatech.edu.*
- *Eli T. Brown is with DePaul University, Chicago, IL, USA. E-mail: eli.t.brown@depaul.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/

attribute level. Users can modify attribute weights and see the resulting ranking of the data points. In the case of multivariate data, ranking systems help people create order from complex data by allowing users to define attribute weights based on how important they believe each attribute is. The data points are then ordered so that the highest ranking data are those that most reflect the significant attributes specified by the user. Specifying attribute weights directly is a flexible and effective way to produce a ranking of data points. Systems like ValueCharts [10] and LineUp [22] allow users to easily visualize the results of rankings based on manually defined attribute weight vectors. Further, they allow users to adjust attribute weights and see how the data point rankings change. Current ranking systems create an effective model of a user's preferences when they know precisely how important attributes are to their decision and remain agnostic to the data points themselves.

However, people often have a better understanding of the holistic relationships between data points as opposed to the absolute importance of attributes. That is, they might have an understanding of relationships between data points but not know what attributes of the data drive that understanding. For example, a person may prefer a Toyota Camry over a Honda Civic but not know how to quantify the importance of fuel economy, horsepower, and price. This problem becomes particularly relevant as the number of attributes that define a dataset increase, and choosing which to emphasize and de-emphasize becomes more difficult and cumbersome. Alternatively to quantifying such preferences, Carterette et al. showed that people are cognitively skilled at making relative judgments on data points (in this case, documents) [11]. When using ranking systems that require quantifying attribute weights, users could consequently find themselves tweaking the attribute weights in order to move a particularly favored data point up in the ranking.

In this paper, we take a first step toward creating a model of attribute weights based on users' subjective preferences in the context of multi-attribute ranking systems. While previous systems allow users to adjust attribute weights and see changes in the rankings to better understand the data, we address a different user task. Our work focuses on creating a technique to help users understand their subjective preferences as it relates to the data attributes. Hence we focus on leveraging a user's cognitive skills at making relative value judgments about data points. We do this by inferring meaning from user interactions. Interaction is a key component in visual analytics, facilitating the understanding of potentially large and complex data. However, its utility is greater than simply advancing a visualization from one state to the next. User interactions form an external capture of the user's cognition [16, 36] and can be used to infer a great deal about the user's analytic process [34] and cognitive state [6]. Thus, important to our goal of capturing user preferences in an analytic model, we view interaction as more than a facilitator for changing the visual state of the system. That is, a user's interactions are motivated by their goals and influenced by their perceptual and cognitive processes, from which we can derive meaning. Interaction is itself data.

We present a prototype system, Podium, to demonstrate our technique for allowing users to specify a subset ranking of data points, from which corresponding attribute weights are computed and visualized. To compute the attribute weights, we apply Ranking SVM [26], using constraints generated from user interactions. As a result, the computed model is used to rank the full dataset. We discuss two potential usage scenarios for this technique: (1) illuminating users' preferences at the attribute level, and (2) deconstructing existing rankings.

The primary contributions of this work include:

1. A multi-attribute ranking prototype, Podium, that weights attributes based on users' subjective preferences for data points
2. A technique for generating constraints for Ranking SVM based on user interactions
3. Two usage scenarios to demonstrate how the technique works

In Section 2, we discuss the foundations of this work with regard to multi-attribute ranking visualization techniques, mathematical and machine learning approaches for ranking, and mixed initiative visual analytics. Section 3 discusses the notion of modeling a user's preferences based on their interactions, and Section 4 describes the interface of Podium and the techniques used to create a ranking based on the user's interactions. In Section 5, we describe two usage scenarios for the system using a 2014 college football dataset. In Section 6, we describe preliminary feedback from users who were given a ranking task to perform using Podium. Section 7 details some of the open questions and challenges we encountered, and Section 8 summarizes the implications of this work.

2 Related Work

In this section, we describe prior work relevant to the challenge of our holistic mixed-initiative ranking approach, including previous multi-attribute ranking visualization techniques (Section 2.1), machine learning approaches to ranking (Section 2.2), mathematical techniques for ranking (Section 2.3), and concepts and principles of mixed-initiative visual analytics (Section 2.4).

2.1 Multi-Attribute Ranking Visualizations

Many systems exist for the visualization of multi-attribute rankings. Spreadsheet programs like Microsoft Excel or Numbers for Mac are commonly used for rankings by applying custom formulas to evaluate tabular data. These systems focus on providing capabilities to generate, modify, and present tabular data and are not intended solely for ranking. Rows (data points) can be sorted by some column (attribute), but such systems do not natively support sorting based on combinations of columns or attributes. Consequently, these systems require a high level of formalism to define a ranking.

Many systems provide interactive interfaces especially for ranking. TableLens is a specialized multi-attribute data visualization technique that utilizes a fisheye technique to display large amounts of data in a single view [37]. Similarly, ValueCharts [10] and LineUp [22] allow users to create custom rankings by clicking and dragging columns to interactively adjust the attribute weights used for the ranking. Users are able to quickly see how changing the attribute weights affect the ranking of the data points. However, these systems still require users to specify attribute weights to produce a ranking of data points.

Other systems were designed for more domain-specific ranking tasks. For example, some systems generate rankings while augmenting the visualization for time series data [42, 45]. di Sciascio et al. applied ranking visualization techniques to relevance scores of document search results [14]. Behrisch et al. proposed an approach to visualize multiple rankings using a small multiples approach [3]. Their technique utilized radial node-link glyphs to visualize the similarities and differences between different rankings.

Our approach differs from previous work in that it allows users to generalize their knowledge and opinions about a subset of data points and apply it to rank the complete dataset. Users are not required to quantify the importance of attributes; instead, they rank order data points, and the system infers the respective attribute weights. The resulting set of attribute weights thus represents a model of the user's subjective preferences.

2.2 Machine Learning

Machine learning has been used in visual analytic systems for various purposes [18, 40]. One purpose is information retrieval which refers to locating information relevant to a particular problem or query, and is analogous to our generalized ranking problem. Based on the user's exemplary rankings, we are interested in finding and ranking relevant data. Metric learning has been used to address information retrieval problems [31, 33], wherein distance functions are learned so that similar documents are near each other while dissimilar documents are far apart. While it can be used to provide an ordered list of distances from a query, metric learning approaches do not inherently have a notion of *order*; therefore, while similar documents may appear near one another, their relative order may not be preserved.

Learning to rank is another well-explored problem defined in machine learning. Learning to rank is typically approached from one of three perspectives: (1) pointwise, (2) pairwise, or (3) listwise [32].

Pointwise learning to rank methods reduce ranking to a regression problem [13], where individual data points are used to train a model. Compared to pairwise and listwise approaches, Liu found that pointwise regression-based approaches performed consistently worse on ranking benchmark document retrieval data [32]. Liu [32] and Cao et al. [9] advocate for listwise approaches, models trained using a full input list of data. For our particular problem, however, listwise approaches would not be appropriate since they rely on being trained with fully ranked lists. To simplify and generalize multi-attribute rankings, the user should be able to rank a handful of data points, and the system should compute the rest. Thus, we chose to use a pairwise approach, where the model is trained using pairwise constraints derived from the subset of data the user ranked.

Several pairwise learning-to-rank algorithms exist, including neural network approaches like RankNet [7] and FRank [44], boosting approaches like RankBoost [19] and AdaBoost [20], and SVM approaches like Ranking SVM [26] and IR-SVM [8]. Of the pairwise learning-to-rank approaches, Liu found that Ranking SVM was one of the most effective [32]. Thus, we have chosen to use Ranking SVM as the underlying model in our prototype for deriving attribute weights from user interactions. We discuss the alternatives to Ranking SVM in more detail in Section 7.

2.3 Mathematical Models for Multi-Attribute Rankings

Many computational approaches exist in decision theory literature for ranking items in a multi-attribute dataset. Zanakis et al. performed a comparison of several methods for multi-attribute decision-making (MADM) [48]. Among those methods were Simple Additive Weighting (SAW), Multiplicative Exponent Weighting (MEW), Analytic Hierarchy Process (AHP), Elimination and Choice Expressing Reality (ELECTRE) [38], and Technique for Preference by Similarity to the Ideal Solution (TOPSIS) [30]. These techniques mathematically evaluate multi-dimensional data points using a set of criteria (attributes) and their respective levels of importance (weightings). Most commonly, people rank data points using a simple additive weighting approach. That is, they apply some weight to each attribute of a dataset, where the weight is a representation of the given attribute's importance to the decision. The weights of the attributes sum to 1, intuitively showing that each attribute comprises part of the whole decision. A score is computed for each data point based on the summation of the weighted attribute values. Data points are then ranked based on the highest computed score.

Podium utilizes a SAW model to compute the ranking of data points. A SAW model is intuitive for users to understand and serves as a well-explored technique often used as a benchmark against which other techniques are evaluated [48]. Further, as described in Section 4.2, the underlying model we use, Ranking SVM, ultimately uses a SAW model to determine the class or order of input data.

2.4 Mixed-Initiative Visual Analytics

Visual analytic technologies invoke cognitive processes and tasks that help people think about data in the context of the world and phenomena in it [27, 43]. For example, many visual analytic techniques have been proposed to help people perform the general activity of sensemaking [2, 35, 39]. This high-level activity consists of gathering data, understanding relationships between the data, comparing the knowledge gained from the exploration with one's own understanding of the world, forming hypotheses about information, and ultimately presenting insights to share with others.

More recently, there are examples of visual analytic systems that adhere to design principles of mixed-initiative systems [25]. Horvitz presented a set of principles to consider for mixed-initiative visual analytics, including ensuring that adding automation increases real value and considering capabilities for systems to continue to learn over time. These principles advocate for a balance of effort between human operators and machines, to promote a collaborative joint-system for the purpose of solving a common goal or task. Thus, mixed-initiative visual analytics focuses on balancing human and machine effort while performing a visual data exploration task. This balance is important, as 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.

Such mixed-initiative systems leverage users' interactions with data to infer their intentions and better inform analytical models. One such system is Dis-Function, a technique presented by Brown et al. that leverages automation to learn distance functions [5]. The approach works by allowing users to directly adjust data points' relative positions on a scatterplot. The system uses these interactions to create models that are tuned to the user's understanding of relationships between the data. Mixed-initiative approaches have also been applied to dimension reduction techniques in scatterplots [4, 17, 28, 29], to the inference of a user's intent in creating a visualization [41], and to make data recommendations based on sensemaking activities [12]. In general, these examples of mixed-initiative systems leverage a small sample of user interactions, approximate an analytical model, and use that model to organize and visualize the remaining data.

Our prototype system, Podium, employs mixed-initiative principles in order to provide users with powerful mathematical models that describe their preferences without the requirement that users be experts in defining and parameterizing such models. With this approach, users can directly manipulate data points by clicking and dragging them to new positions in the table. The system then provides automated modeling of those interactions to produce an attribute weight vector that can enlighten the user about which attributes contribute to their preferences. While leveraging similar principles to balance human and machine effort, we build on previous work by demonstrating how mixed-initiative approaches can benefit users of multi-attribute ranking systems to model their subjective preferences.

3 Modeling User Preferences

In previous work on ranking data points, systems allow users to modify attribute weights to see how the ordering of the data is affected. Our proposed technique focuses instead on understanding how a user's subjective preferences and biases materialize in their interactions with data. While recent work has focused on detecting and mitigating cognitive bias in visual analytics [15, 21, 46, 47], Green et al. note that systems employed to counteract human biases can limit people's ability to create effective mental models [23]. We confront this gap by utilizing a user's interactions to create a model of the data that better reflects their cognitive perceptions. Specifically, we derive an attribute weight vector based on the user's interactions with the data using a Ranking SVM [26] model. This weight vector represents an external capture of the user's mental model of the relative value of data points, allowing users to reflect on the accuracy of their subjective preference (or bias) towards the attributes used to rank data points. The resulting weight vector is then used in a Simple Additive Weighting (SAW) model [48] to produce the full ranking of data points.

This approach to ranking may lead users to simply confirm their existing biases about the data; however, the primary purpose of the system is to allow such an exploration of the data in light of subjective preferences or perceived value. By doing so, users can 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, it may derive a set of attribute weights that ultimately results in the preferred data point being ranked below others that the user interacted with. The user might then conclude that based on their perception of the data, it simply is not possible to derive a model in which the data point is the top ranked item. Alternatively, the system may be able to compute a model, but it may result in attribute weights that do not fit the user's mental model. For example, consider a user ranking college football teams so that their favorite team is on top. The system might compute a model where offensive average points per game is given negative weight. However, having a greater number of points seems like it should be a positively weighted attribute that contributes to a team's success; thus, the user might conclude that although they love their favorite team, it may not objectively be very good. Thus, our

proposed technique helps users to become more aware of how their perception of the relationships between the data points may or may not be grounded in the data.

4 PODIUM

In this section, we describe Podium, a prototype multi-attribute ranking system that models user interactions with data points. Podium lets users demonstrate a preferred ranking of a subset of data points, from which the user's preferences are modeled and the remaining data points are ranked to follow these preferences. In Section 4.1, we discuss the characteristics of the Podium interface in more detail. In Section 4.2, we discuss the underlying techniques used by the system to model a user's preferences, and in Section 4.3, we discuss how we visualize the contribution of each attribute to a data point's ranking.

4.1 User Interface

The Podium interface is divided into two components: the *main table* (Figure 1a) and the *control panel* (Figure 1b). The main table is the main data view of the system. It shows a table where each row is a data point and each column represents an attribute. The data are ordered by their ranking, and initially the rank scores are computed assuming all attributes are equally important (see Section 4.2.3 for how). Users can interact with the data in the table by clicking and dragging rows to a new position. The control panel contains visualization and model controls including the *Compute Weights* button, which causes the system to derive a new set of attribute weights based on the user's interactions with the rows in the table. Pressing the *Rank* button will apply the attribute weights to the dataset, ranking all of the rows in the table according to their resulting rank scores.

4.1.1 Main Table

The main table contains the full set of data points and attribute values (Figure 1a). The values in the three left-most columns are not part of the loaded dataset, but are important values computed as the user interacts with the system. The leftmost column contains the *Absolute Rank* for the given data point. The table is always sorted by the Absolute Rank column. The *Relative Rank* column shows the relative position of each training row with respect to the other rows used to train the underlying model. After the weight model is applied to the table, the Relative Rank column shows the new relative position of the row as well as the previous user-defined relative position (Figure 1c). This helps users to see which constraints they specified were preserved by the model and which were violated. The *Rank Score* column contains bars that encode the score computed for each data point. Bars with a larger width have a higher score than bars with a smaller width. The computation for the rank score is described in Section 4.2.3. The exact score encoded in the bars in the Rank Score column can be seen in a tooltip by hovering over the bar.

Bar Overlay. Attribute values can be encoded as the width of bars with the encoded values overlayed as text by checking the *Show Bars* option in the control panel. The value of each attribute is encoded as the width of the bar, making it easier to spot trends within the table. This is similar to the technique used in TableLens to give people a quick overview of data attributes and values [37].

Attribute Contribution. Each cell containing the value of an attribute can be augmented to visualize the individual contribution of each attribute to the data point's rank score using the *Show Contribution* option from the control panel. The contribution value corresponds to how much of the data point's rank score is the result of the given attribute value. Comparing an attribute's actual value to its contribution can be enlightening to the user. For example, a data point may have an exceptionally high value for a particular attribute; however, if the magnitude of the weight of that attribute is low, then the attribute may not have much impact on the ranking of the data point. Contribution values are visualized as thin vertical black bars, where the horizontal position within the cell is a normalized value between 0 and 1, where larger values correspond to a greater contribution of that attribute to the data point's rank score. The computation for the contribution is explained further in Section 4.3.

Interactions. There is one primary interaction in the interface: clicking and dragging rows (i.e., data points) to demonstrate the user's preferred ranking of a subset of the data points. The rows within the table can be moved to a new position by click and drag. When a row is moved to a new position, the row is colored according to how the absolute rank changed. Red indicates that the row moved down, and green indicates that the row moved up. The opacity of the color indicates how far the row moved; lower opacity means that a row moved less, while higher opacity means that a row moved farther. Rows in the table may also be marked by clicking on the row. This changes the background color of the absolute and relative rank cells to gray. Rows that the user moved to a new position are marked automatically. Marked rows are visually distinct and can thus be easily tracked as they are re-ranked or re-positioned. All marked rows are ultimately used to generate training tuples for SVM.

4.1.2 Control Panel

The control panel has three components: visual controls for the main table (Figure 2a), attribute or minimap view (Figure 2b), and model controls (Figure 2c).

The interface has three toggles for **visual encodings** in the main table: *Show Change*, *Show Bars*, and *Show Contribution* (Figure 2a). Show Change, when checked, colors the rows in the table that moved as a result of the user's interactions: red when the row moves down and green when the row moves up. For example, moving a data point from row 2 up to row 1 would result in the moved data point being colored green, while the point previously in row 1 would be colored red since it moved down to row 2. This coloring scheme applies to both the main table and the minimap. Show Bars and Show Contribution show or hide the visual encoding of attribute values as bars in the table and the vertical bars representing the contribution of each attribute to the data point's rank score, respectively. Figure 3 shows what the interface looks like when all of the visual encoding toggles are activated.

The interface has two **additional views**: *attributes* and *minimap* (Figure 2b).

Fig. 2: The control panel is comprised of visual encoding controls (a), additional views of the attribute weights (b), and controls for the underlying models (c).

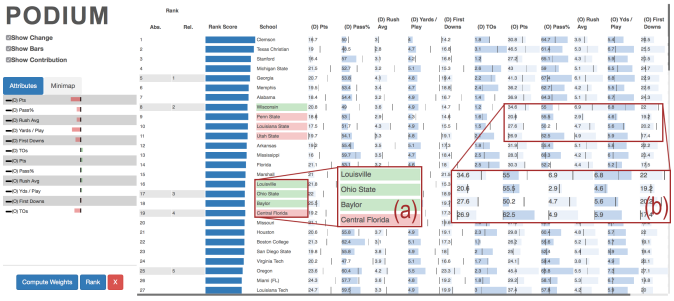
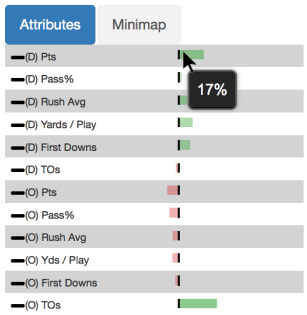
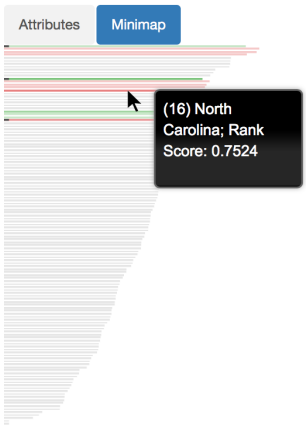


Fig. 3: This figure shows the main table when the visual encoding options Show Change, Show Bars, and Show Contribution are selected in the control panel. (a) In addition to changing the color of the row the user moved down to red (row 19), Show Change changes the colors of the rows green that also moved up as a consequence (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.



(a) *Attributes* tab encodes the weight for each attribute as the width of the bar in the bar chart.



(b) *Minimap* tab represents each row in the table with a bar that encodes the corresponding rank scores. Dark gray squares to the left of the bar indicate rows in the main table that are marked for training.

Fig. 4: The two tabbed views in the control panel.

Attributes. The attributes tab (Figure 4a) visualizes the attribute weight vector. Each attribute has a corresponding bar representing its weight as a value between -1 and 1 (between -100% and 100%), where bars representing negative weights are red and bars representing positive weights are green. The weights can be directly adjusted by clicking on the bar and dragging it to the left to decrease the weight or to the right to increase the weight. When the Compute Weights button is pressed, the widths of the bars will be updated based on the weights computed by Ranking SVM. We chose this layout for representing attribute weights over a direct column manipulation approach like that used in LineUp [22] due to the inherent difficulties comparing relative attribute values horizontally. The vertical positioning of the attribute weight bars allows users to easily make comparisons between values.

Clicking on an attribute's name toggles between three values. The up arrow (🡅) indicates that the user would like the model to produce a higher weight for the given attribute. The down arrow (🡇) indicates that the user would like the model to produce a lower weight for the given attribute. The default option is a dash (━), which has no effect. The up and down arrows result in additional constraints being added to Ranking SVM, as described in Section 4.2.2.

Minimap. The minimap (Figure 4b) displays a miniaturized version of the main table, where each horizontal bar corresponds to the rank score of a data point in the main table. The colors mirror that of the main table: green indicates a row that moved up, while red indicates a row that moved down. On hover of a bar, a tooltip displays the ranking, the name of the given data point, and the numerical value of the rank score.

The minimap allows the user to see trends and anomalies of the table as a whole at a glance. For instance, the user may be able to easily spot a cluster of points in the middle that all moved up in the ranking or a particularly dark red data point that fell far down in the ranking. They may notice a sharp drop in the widths of the bars after the fifth ranked item, indicating that the top five data points are significantly better than the points below. The minimap bars may also have a small dark gray square to the far left indicating that the corresponding data point is marked in the main table to be used by Ranking SVM in computing the weight vector. This helps the user to easily locate items of interest.

The interface has three buttons for **model controls**: *Compute Weights*, *Rank*, and *Discard* (Figure 2c).

Compute Weights. The Compute Weights button triggers the system to derive a new set of attribute weights based on the user's interactions with the data points in the table. The system models interactions with data points in the main table in order to produce a new set of weights according to the techniques described in Section 4.2.

Rank. The Rank button triggers the system to apply the set of attribute weights to the data points in the table. The weights might have been derived by Ranking SVM, or they might have been manually specified by the user. In any case, the weights are used to compute a score for each data point, which then determines the rank order of the data in the table according to the techniques described in Section 4.2.3. The main table and minimap views are then updated to reflect the newly computed ranking.

Discard. The Discard button, labeled in the interface as *X*, allows a user to return the system to the last saved state. State information including attribute weights, ranking, and marked rows is saved each time the Rank button is pressed.

4.2 Weight Solver

The weight solver, Ranking SVM [26], is triggered by the Compute Weights button. The attribute weights may be positive or negative. A positive weight ($w_j > 0$) indicates that higher values of the attribute w_j are preferred in the ranking, while a negative weight ($w_j < 0$) indicates that lower values of the attribute are preferred in the ranking. A weight of $w_j = 0$ indicates that the given attribute w_j does not impact the ranking of the data points. In the following sections, we first describe how Ranking SVM is used to derive the attribute weight vector. Next, we discuss how the constraints are derived from the user's interactions with the data to train the Ranking SVM model. Finally, we describe how the weight vector is applied to produce a full ranking of the data points. Table 1 summarizes the notations used in this section. In the current version of the system, only numerical attributes are used in the computation of the ranking. However, Ranking SVM can be extended to support categorical attributes, which we describe further in the Discussion section.

4.2.1 Ranking SVM

Podium uses Ranking SVM [26] to derive a set of attribute weights based on the user's ranking of the rows in the table. Ranking SVM applies the classic support vector machine (SVM) intuition to the ranking problem. We have used the implementation of SVM provided by the python library Scikit-Learn [1]. In a standard SVM, the algorithm is provided with data points in some m dimensional space, and a corresponding set of labels, marking to which class each point belongs. This is often represented as a set of tuples, $(\mathbf{d_i}, y_i)$ for data point $\mathbf{d_i} \in \mathbb{R}^m$ and label $y_i \in \{-1, 1\}$ for a two-class problem. The output model is a hyperplane, defined by a vector in $\mathbb{R}^m$, that cuts through the space

[1] http://scikit-learn.org

Notation	Description
$\mathbf{a} = \{a_1, \ldots, a_m\}$	set of m attributes describing D
$\mathbf{w} = [w_1, \ldots, w_m]$	attribute weight vector; $\mathbf{w} \in \mathbb{R}^m$
$D = \{\mathbf{d_1}, \ldots, \mathbf{d_n}\}$	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

of the data, optimized so that points with $y_i = -1$ are on one side of the hyperplane, points with $y_i = 1$ are on the other, and there is 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. Rather than a full set of data points with corresponding labels, we get a limited set of pairs of data points $\mathbf{d_i}$ and $\mathbf{d_j}$, and a label telling us if $\mathbf{d_i}$ is considered better or not. The data Ranking SVM will see are difference vectors for pairs of data points, e.g. $\mathbf{d_i} - \mathbf{d_j}$, and it will be asked to predict which point is better based on their difference [26]. Specifically, we transform the pair $(\mathbf{d_i}, \mathbf{d_j})$ and their relative rank to a tuple according to Equation 1.

$$\left(\mathbf{d_i} - \mathbf{d_j}, \begin{array}{l} 1 \text{ if } \mathbf{d_i} \text{ is preferred,} \\ -1 \text{ otherwise} \end{array}\right) \tag{1}$$

The resulting model using Ranking SVM can be used to input a pair of points and predict which is better. However, all of the constraints derived from the user's interactions might not be satisfiable. Since producing no model might be frustrating to users, we have chosen to model all constraints as soft constraints rather than hard constraints. As such, the user's interactions will always produce a set of attribute weights that models the user's constraints as closely as possible by penalizing constraint violations. In the next section, we describe how the user's interactions are transformed into the input needed by the Ranking SVM.

4.2.2 Deriving Constraints

In order to compute SVM's linear separator, the ranking problem must first be transformed into a two-class classification problem. To do so, we generate labeled data for Ranking SVM using the points with which the user has interacted by dragging to a new position or clicking on. These are the k marked rows in the main display table. These k points have indices $[l_1, \ldots, l_k]$, thus we consider data points $\{\mathbf{d}_{l_1}, \ldots, \mathbf{d}_{l_k}\}$. Within this set, we create the set of all combinations of pairwise difference vectors as training instances. That is, for every $i, j \in \{1 \ldots k\}$, where $i \neq j$, we derive a training tuple according to Equation 1. Intuitively, each training instance is the difference between a pair of rows $\mathbf{d_i}$ and $\mathbf{d_j}$, classified as $y = 1$ if $\mathbf{d_i}$ is ranked higher than $\mathbf{d_j}$, or $y = -1$ if $\mathbf{d_i}$ is ranked lower than $\mathbf{d_j}$. Thus, we now have a two-class classification problem, to which Ranking SVM can be applied.

If the user presses the Compute Weights button in order to derive a weight vector when fewer than $k = 5$ rows are marked, the system automatically marks surrounding rows until $k \geq 5$ have been marked. For example, if the user has only interacted with 2 rows, the system will add the rows above and below those that were interacted with to provide additional training data for Ranking SVM. We chose $k = 5$ to ensure a minimum amount of training data for deriving the attribute weight vector. We make the assumption here that when the user places a data point $\mathbf{d_i}$ at rank position j, they make a value judgment about the surrounding rows. That is, if $\mathbf{d_i}$ is placed at rank j, then the data at $j-1$ is better than $\mathbf{d_i}$ and $\mathbf{d_i}$ is better than the data located at $j+1$. If $k > 5$, no additional training data is added.

Beyond the rows that the user interacted with in the table, the user may also emphasize or de-emphasize an attribute's weight by toggling its associated up (🡅) or down arrow (🡇) in the control panel. In response, the system adds a new $(\mathbf{x}, y)$ tuple to the training instances where $x = [0, \ldots, 0, 1, 0, \ldots, 0] \in \mathbb{R}^m$ is the vector with 0 in all positions except for a 1 in the position representing the attribute the user wishes to affect. If the user intends to emphasize the attribute (🡅), we use a training pair with values $(-\mathbf{x}, y = -1)$, and if the user intends to de-emphasize the attribute (🡇), the training pair is $(\mathbf{x}, y = -1)$. The choices of sign for this technique are derived from the Ranking SVM update algorithm and empirically validated to ensure they can cause a meaningful change to the weight vector.

4.2.3 Ranking

After transforming user inputs as described in Section 4.2.2 and learning a Ranking SVM model as described in Section 4.2.1, we have a weight vector $\mathbf{w}$ representing the model for user preference as a signed importance value for each data attribute. Additionally, the user may adjust these weights as described in Section 4.1.1. This section describes how we rank the data points based on this attribute weighting.

The process of computing a ranking is modeled on the way Ranking SVM's model works. With that model, difference vectors of two data points are passed to the Ranking SVM classifier (e.g., for points i and j, $(\mathbf{d_i} - \mathbf{d_j})$). In response, Ranking SVM uses its model to predict which of the two input points should be ranked higher. Specifically, it uses the dot product of the given difference vector with its internal model, a weight vector $\mathbf{w}$. If the dot product $\mathbf{w} \cdot (\mathbf{d_i} - \mathbf{d_j})$ is positive, the difference vector belongs to the positive class $y = 1$, and thus $\mathbf{d_i}$ belongs relatively before $\mathbf{d_j}$ in the ranking. If the dot product is negative, the difference vector belongs to the negative class $y = -1$, and $\mathbf{d_j}$ belongs relatively before $\mathbf{d_i}$ 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 could be used in this way to derive a ranking, by using its output on any pair of data points as a comparator function in an $O(n \log n)$ sorting routine. However, rather than feeding difference vectors to Ranking SVM to obtain 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. Specifically, we calculate individual dot products of $\mathbf{w}$ with each data point to produce a rank score as in Equation 2.

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

These dot products 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 $\mathbf{w} \cdot (\mathbf{d_i} - \mathbf{d_j}) = \mathbf{w} \cdot \mathbf{d_i} - \mathbf{w} \cdot \mathbf{d_j}$. If $\mathbf{w} \cdot \mathbf{d_i} > \mathbf{w} \cdot \mathbf{d_j}$, then $\mathbf{w} \cdot (\mathbf{d_i} - \mathbf{d_j}) > 0$ so $\mathbf{d_i}$ is ranked above $\mathbf{d_j}$, and vice versa. This alternative approach allows us the flexibility to use the same ranking algorithm for the points whether the weights come from Ranking SVM or the user has modified them directly via the interface.

4.3 Visualizing the Contribution

The contribution value for an attribute of a data point helps a user to understand how much of that data point's rank score, $r(\mathbf{d_i})$ (see Equation 2), is due to the given attribute. The Contribution score, $\hat{C}(d_{ij})$, represents the normalized contribution of attribute a_j for the rank score of data point $\mathbf{d_i}$. $\hat{C}(d_{ij})$ is computed based on the proportion of the rank score that comes from $d_{ij} \cdot w_j$, described in Equation 3.

$$\hat{C}(d_{ij}) = \frac{|d_{ij} w_j|}{\max_l \{|d_{il} w_l|\}} \tag{3}$$

The result is $\hat{C}(d_{i,j}) \in [0, 1]$ where higher values indicate that the attribute contributes more to the data point's rank score. When an attribute weight has a high magnitude, the attribute values will have a higher contribution to the data point's rank score. When an attribute

weight has a low magnitude, the attribute values will have a lower contribution to the data point's rank score. The largest contribution is when both the attribute weight and the data value have high magnitude. While a visually and computationally simple feature, this allows users to easily understand which attributes are important to the ranking of a data point. Figure 3b shows the contribution values for a portion of the table with a dataset of college football teams. Wisconsin (row 8), for instance, has fairly high offensive statistics; however, since offensive statistics are not weighted very highly in the model, they contribute a relatively small amount to the rank score (as seen by the position of the vertical black bars).

5 Usage Scenarios

In this section, we present two usage scenarios, each illustrating one way that our interactive, automatic ranking system can be used. In the first case study (Section 5.1), a sports fan, Grace, uses a dataset about college football to discover the most important features of her favorite teams. In our second scenario (Section 5.2), using the same dataset, another football fan, Ada, uses Podium to discover variables that will help her predict the success of teams in upcoming seasons.

5.1 Illuminating User Preferences

Grace is a college football fan from Georgia. She wants to use Podium to understand which statistics make her favorite teams successful. She uses a college football dataset from 2014 [2] that contains entries for 128 different schools and contains 12 numerical attributes representing defensive and offensive statistics, including points, passing percentage, rushing average, yards per play, first downs, and turnovers. She begins by positioning a few of her favorite teams in the table. She drags Georgia and Florida State up near the top of the table. Auburn is a good team as well, but not quite as good as Georgia and Florida State, so she positions Auburn a bit lower. She does not think Georgia Tech and New Mexico State are very good teams, so she drags them below the other teams. She presses Compute Weights to compute an attribute weight vector based on her constraints for the teams. Figure 5a depicts this stage of Grace's exploration, annotated to show the rows she interacted with and the resulting attribute weights.

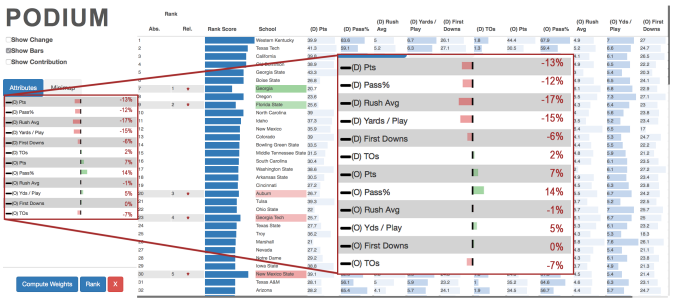
Next, she applies the weight vector to the table by pressing Rank. The results are shown in Figure 5b. The relative order of the teams is maintained in the ranking, with Georgia at #5, Florida State at #50, Auburn at #56, Georgia Tech at #96, and New Mexico State at #125.

Grace notices that Alabama has been positioned at #4 based on the weight vector defined by Ranking SVM. Alabama is a great team, better than Georgia, she thinks. She clicks on the row to refine the model and reinforce Alabama's correct positioning to the ranking model. In the control panel, she notices most of the offensive statistics are weighted positively except for rushing average. Lots of rushing yards per play should be beneficial for a team, so she toggles the arrow up (↑) to emphasize the attribute in the model. Figure 5c depicts this phase of Grace's exploration.

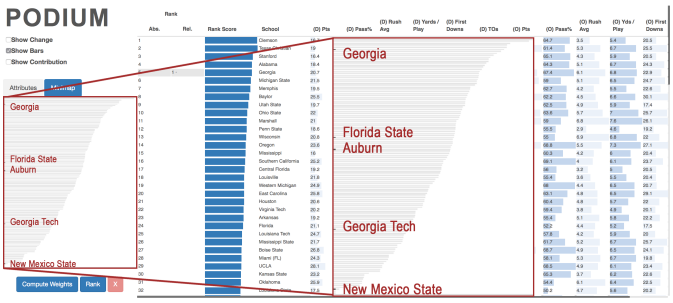
Grace presses Compute Weights to regenerate the model using the new constraints and presses Rank to apply the resulting weight model to the full dataset. The final model increased offensive rushing average from -1% to 7% as well as slightly shifted some of the other attributes' weights. Since a negative attribute weight indicates that low values of an attribute are more valued, Grace recognizes that 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. Grace toggles the Show Contribution option in the control panel. She sees that while Alabama and Georgia are both very strong for offensive statistics, defense contributes far more to the success of her favorite teams (Figure 5d).

From her exploration, Grace realized that her favorite teams were successful due to particularly strong defenses that allowed very few points, complete passes, and rushing yards. Offense was generally less

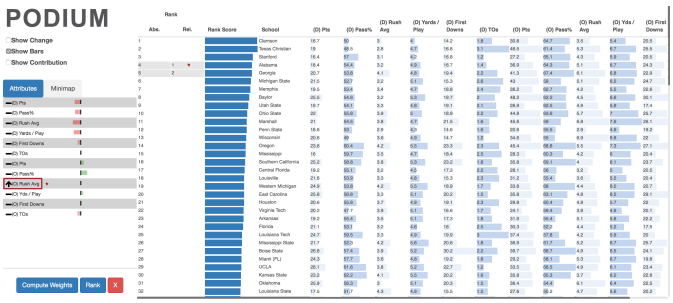
[2] http://www.sports-reference.com/cfb/



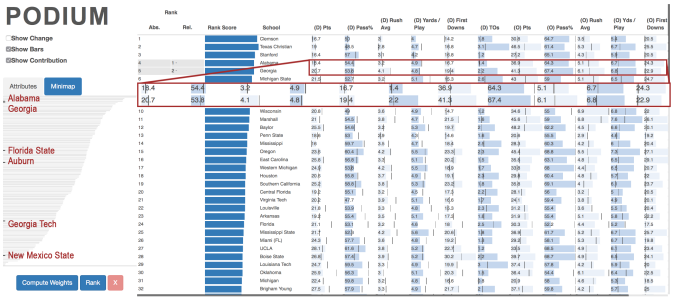
(a) Step 1: A view of Grace's interactions with the main table, annotated by (*). She moved rows representing Georgia, Florida State, Auburn, Georgia Tech, and New Mexico State. The resulting attribute weights from Ranking SVM are shown in the red callout box.



(b) Step 2: The result of ranking using the weight vector in (a). The locations in the table of the teams Grace interacted with are annotated in the minimap.



(c) Step 3: A view of Grace's interactions, annotated by (*). She marked Alabama (to indicate she is happy with the rank) and toggled offensive rushing average to (↑) to emphasize the attribute.



(d) Step 4: A view of the final configuration from Grace's exploration. The locations of the teams she interacted with are annotated in the minimap on the left and the contribution visualization is shown in the red callout box.

Fig. 5: A series of screenshots of the system to support the usage scenario described in Section 5.1. Red annotations show relevant areas of the display.

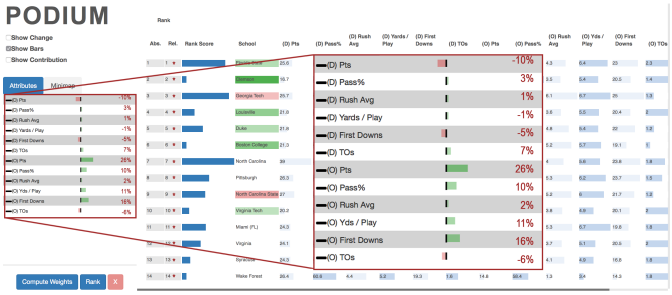
important to the success of Grace's favorite teams, with the exception of passing percentage at 14%.

5.2 Deconstructing Existing Rankings

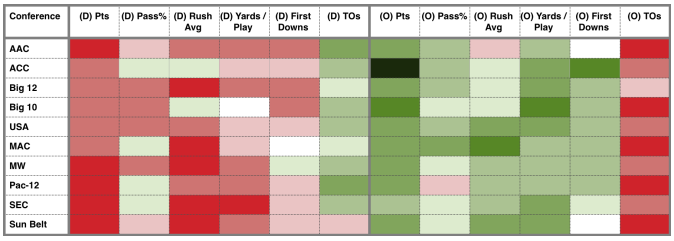
Ada is interested in using Podium to understand what made teams successful in the final 2014 standings. She wants to understand what the winning teams did well or poorly so she can make better predictions about successful teams in the future. She compares the teams from the same 2014 college football dataset as her friend Grace, ranking them according to their final standings from the end of the season.

Ada begins by focusing on the subset of 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. She drags the teams into their rank positions in the table and presses Compute Weights (Figure 6a). Podium responds by showing a set of weights of the team attributes that can explain that ranking. She finds that ACC teams that are successful rely mostly on offense, and in particular, they rely heavily on offensive points and first downs (in contrast to some other conferences where teams with strong defense perform well).

Interested in seeing how the model may differ for each conference, Ada performs a similar task on a subset of the data for each conference. She exports the weights of the attributes to visualize her results (Figure 6b). Weights range from -15% to +26%. They are color-coded so that negative weights are red and positive weights are green. Percentages are binned in increments of 5%. She finds that many conferences rely very heavily on offense as indicators of success, placing more than 60% weight on offensive statistics (ACC, Big 10, USA, and MAC). Success in other conferences relies more than 50% on defensive statistics (AAC, Big 12, and SEC). She finds that ACC relies far more heavily on offensive points as indicators of success than any other conference, making up 26% of the model with a single statistic. She also finds a few outliers: AAC has negative weight for offensive rushing average; SEC has negative weight for offensive passing percentage; and Sun Belt has negative weight for defensive turnovers. Similarly, several conferences have positive weight for defensive passing percentage, defensive rushing average, and defensive first downs. Based on what she's learned, Ada can use the weight model from each conference in 2014 along with other seasons to understand trends and better predict team performance in the coming seasons.



(a) Step 1: A view of Ada's interactions in ranking the teams of the ACC and the resulting attribute weight vector in the red callout box.



(b) Step 2: Ada exports the attribute weights to create a heat map for each conference's final 2014 rankings. The weights range from -15% to +26%. Negative attribute weights are red, while positive attribute weights are green. The opacity of each cell encodes the magnitude of the weight.

Fig. 6: A series of screenshots of the system from the usage scenario described in Section 5.2.

6 Preliminary User Feedback

We conducted a session to gather preliminary user feedback from 4 participants (P1-P4) who are experts in information visualization. The goal of the session was to obtain preliminary feedback on the attribute weight model as well as the general usability of the prototype system. Participants were first given a short tutorial of how the system works, during which they were encouraged to ask questions as needed. Next, they were given 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 [3] and reflect on the model results and interface presentation. Users were asked to think aloud throughout the session. Below we summarize the participants' perspectives on some of the strengths and weaknesses of our approach.

All participants believed the holistic approach to ranking was very useful and 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). Given this preliminary feedback from users, we discuss the implications in greater detail in the next section.

7 Discussion and Limitations

7.1 Interface

Relative Comparisons. The interface of Podium provides an intuitive way for users to interact with the data. However, it requires users to place data points at precise rankings in the table. In the same way that

[3] https://www.kaggle.com/deepmatrix/imdb-5000-movie-dataset

users may have a difficult time quantifying how important an attribute is with direct attribute-manipulation ranking systems, they may have a hard time deciding where to place a data point. Consequently, the technique we chose to use to model the attribute weight vector (Ranking SVM) only models the relative positions of data points. 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. Thus, future work might include understanding which interface alternatives are most intuitive in helping users to make relative comparisons of data.

Explainability. Our techniques allow a user to see how their holistic preferences for data points translate to preferences at the attribute level. Podium facilitates the comparison of users' expectations with reality and can thus help users identify inconsistencies or biases in their perception of data. Based on preliminary user feedback, it is apparent that 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. Thus, 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. That is, when a data point that the user moved ends up in an unexpected position, how can the system explain the model to the user? Further, could we provide some measure for how biased a user is based on how their perceptions may or may not be grounded in the data?

7.2 Ranking Solver

Inferring Constraints. In order to train SVM, we rely on deriving constraints based on the user's interactions with the rows in the table. It remains an open question to understand to what extent the user considers the implicit ordering of data when re-positioning a row in the table. For example, if the user places a data point at row j, perhaps the user implicitly decided that the data at row j was better than the data at row $j+1$. The user may also have decided that the data at row j was not as good as the data at row $j-1$. They may have considered both, or they may have considered neither. Thus, we defer to future work the task of understanding which implicit decisions are made based on a user's interactions. We currently employ a conservative approach whereby constraints are derived only using the rows that the user explicitly interacted with.

SVM Alternatives. Ranking SVM is an effective model for pairwise learning to rank [32]. We chose an approach that is sensitive only to the relative order of data points for the same reason that users may find traditional ranking systems difficult: in the same way that users may find it difficult to quantify the importance of attributes, they may find it difficult to accurately position data points at an absolute position. However, other pairwise learning to rank approaches exist including RankNet [7], RankBoost [19], and IR-SVM [8], 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 as in Section 5.2, 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.

Model Quality. Given our current choice of Ranking SVM, it is also important to understand the quality of the resulting model. Based on preliminary observations, it is often the case that users interact with Podium for short periods of time resulting in relatively small amounts of data available for training SVM. Consequently, the model can suffer from underfitting compared to a user's "ground truth" of what the ranking of the data should be. This can be improved with more interaction; that is, the user can provide additional constraints in the form of marking or re-ordering more rows in the table to improve the fit of the resulting model. Another improvement to the model fit might result from future work to understand how additional constraints might be derived from a single interaction from the user. If a user drags a data point to a new position in the table, do they make a value judgment about the data points above and below the new position? What about two spots above or below the new position? By understanding which surrounding data the user considers when positioning a data point in the ranking, we can potentially derive additional constraints and further improve the fit of the model.

7.3 Limitations

Carterette et al. discuss the ability of humans to skillfully make comparisons between two items [11]. We have used this previous work as motivation for our choice of an underlying model that relies on relative positions of data points. However, it is unclear to what extent the idea holds when extended from document sets to potentially large multi-attribute datasets. Potential future work could include an exploration of humans' cognitive abilities to make such judgments in the context of multi-attribute ranking systems.

One additional limitation of the current implementation of Podium is its reliance on purely numerical data. Categorical attributes are not presently supported. However, Ranking SVM can be extended to handle categorical data by creating separate binary attributes for each categorical value. For example, if the color attribute has three potential values (red, blue, and yellow), each color value can be transformed into a binary attribute. Any data point that had the value red for the color attribute would have a 1 for the red attribute and 0 for the blue and yellow attributes in the transformed space.

8 Conclusion

Ranking data points is a commonly-performed task for many data-driven decisions. Previous ranking systems require users to manually specify attribute weights of their dataset to produce system-generated rankings. As dataset sizes and complexities increase, this can become a cognitively difficult task. Often, people have a better understanding of data points as a whole than they do of the attributes that define them. In this paper, we thus introduced a technique to create a ranking of multi-attribute data where users are able to rank a subset of data points, from which the system computes a descriptive set of attribute weights and produces a new ranking of the data. The technique uses Ranking SVM trained on the data points that the user has interacted with in order to determine the weight of each attribute. To demonstrate this technique, we described two usage scenarios with the prototype system, Podium, and presented preliminary user feedback.

Acknowledgments

Support for the research is partially provided by the DHS VACCINE Center of Excellence, the U.S. Department of Energy through the Analysis in Motion Initiative at Pacific Northwest National Laboratory (PNNL), DARPA FA8750-17-2-0107, and the Department of Defense. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

References

[1] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*, pages 111–117, 2005.

[2] S. J. Attfield, S. K. Hara, and B. W. Wong. Sensemaking in Visual Analytics: Processes and Challenges. pages 1–6. The Eurographics Association, 2010.

[3] M. Behrisch, J. Davey, S. Simon, T. Schreck, D. Keim, and J. Kohlhammer. Visual Comparison of Orderings and Rankings. *EuroVis Workshop on Visual Analytics*, 2013.

[4] J. Broekens, T. Cocx, and W. A. Kosters. Object-centered interactive multi-dimensional scaling: Ask the expert. *Proceedings of the 18th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*, pages 59–66, 2006.

[5] 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.

[6] E. T. Brown, A. Ottley, H. Zhao, Q. Lin, R. Souvenir, A. Endert, and R. Chang. Finding waldo: Learning about users from their interactions. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1663–1672, 2014.

[7] 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.

[8] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193. ACM, 2006.

[9] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to Rank: From Pairwise Approach to Listwise Approach. *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136, 2007.

[10] G. Carenini and J. Lloyd. ValueCharts: Analyzing Linear Models Expressing Preferences and Evaluations. *Working Conference on Advanced Visual Interfaces (AVI)*, pages 150–157, 2004.

[11] 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.

[12] K. Cook, N. Cramer, D. Israel, M. Wolverton, J. Bruce, R. Burtner, and A. Endert. Mixed-initiative visual analytics using task-driven recommendations. In *Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on*, pages 9–16. IEEE, 2015.

[13] D. Cossock and T. Zhang. Subset Ranking Using Regression. *COLT*, pages 605–619, 2006.

[14] C. di Sciascio, S. Vedran, and E. E. Veas. Rank As You Go: User-Driven Exploration of Search Results. *21st International Conference on Intelligent User Interfaces, IUI 2016*, (May):118–129, 2016.

[15] E. Dimara, A. Bezerianos, and P. Dragicevic. The attraction effect in information visualization. *IEEE transactions on visualization and computer graphics*, 23(1):471–480, 2017.

[16] W. Dou, D. H. Jeong, F. Stukes, W. Ribarsky, H. R. Lipford, and R. Chang. Recovering Reasoning Process From User Interactions. *IEEE Computer Graphics & Applications*, May/June(March):52–61, 2009.

[17] A. Endert, C. Han, D. Maiti, L. House, S. C. Leman, and C. North. Observation-level Interaction with Statistical Models for Visual Analytics. In *IEEE VAST*, pages 121–130, 2011.

[18] A. Endert, W. Ribarsky, C. Turkay, B. Wong, I. Nabney, I. D. Blanco, and F. Rossi. The state of the art in integrating machine learning into visual analytics. In *Computer Graphics Forum*. Wiley Online Library, 2017.

[19] 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.

[20] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of online learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[21] D. Gotz, S. Sun, and N. Cao. Adaptive contextualization: Combating bias during high-dimensional visualization and data selection. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 85–95. ACM, 2016.

[22] 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.

[23] T. Green, W. Ribarsky, and B. Fisher. Building and applying a human cognition model for visual analytics. *Information Visualization*, 8(November 2008):1–13, 2009.

[24] E. Hetzler and A. Turner. Analysis experiences using information visualization. *IEEE Computer Graphics and Applications*, 24(5):22–26, 2004.

[25] E. Horvitz. Principles of Mixed-Initiative User Interfaces. *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, (May):159–166, 1999.

[26] 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.

[27] A. Kerren, J. Stasko, J.-D. Fekete, C. North, D. Keim, G. Andrienko, C. Görg, J. Kohlhammer, and G. Melançon. Visual Analytics: Definition, Process, and Challenges. In *Information Visualization*, volume 4950 of *Lecture Notes in Computer Science*, pages 154–175. Springer Berlin / Heidelberg, 2008.

[28] 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.

[29] 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.

[30] Y. J. Lai, T. Y. Liu, and C. L. Hwang. TOPSIS for MODM. *European Journal of Operational Research*, 76(3):486–500, 1994.

[31] J.-e. Lee, R. Jin, and A. K. Jain. Rank-based Distance Metric Learning: An Application to Image Retrieval. *Computer Vision and Pattern Recognition*, 2008.

[32] T. Liu. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2007.

[33] B. Mcfee and G. Lanckriet. Metric Learning to Rank. *Proceedings of the 27th International Conference on Machine Learning*, pages 775–782, 2010.

[34] C. North, R. May, R. Chang, B. Pike, A. Endert, G. A. Fink, and W. Dou. Analytic Provenance: Process + Interaction + Insight. *29th Annual CHI Conference on Human Factors in Computing Systems, CHI 2011*, pages 33–36, 2011.

[35] P. Pirolli and S. Card. Sensemaking Processes of Intelligence Analysts and Possible Leverage Points as Identified Though Cognitive Task Analysis. *Proceedings of the 2005 International Conference on Intelligence Analysis, McLean, Virginia*, page 6, 2005.

[36] E. D. Ragan, A. Endert, J. Sanyal, and J. Chen. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *IEEE transactions on visualization and computer graphics*, 22(1):31–40, 2016.

[37] 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.

[38] B. Roy. The outranking foundations aproach and the methods. *Theory and Decision*, 31:49–73, 1991.

[39] D. M. Russell, M. J. Stefik, P. Pirolli, and S. K. Card. The cost structure of sensemaking. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 269–276. ACM, 1993.

[40] D. Sacha, M. Sedlmair, L. Zhang, J. A. Lee, D. Weiskopf, S. North, and D. Keim. Human-centered machine learning through interactive visualization. ESANN, 2016.

[41] B. Saket, H. Kim, E. T. Brown, and A. Endert. Visualization by demonstration: An interaction paradigm for visual data exploration. *IEEE Transactions on Visualization & Computer Graphics (InfoVis)*, 2017.

[42] C. Shi, W. Cui, S. Liu, P. Xu, W. Chen, and H. Qu. Rankexplorer: Visualization of ranking changes in large time series data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2669–2678, 2012.

[43] J. J. Thomas and K. A. Cook. Visualization Viewpoints: A Visual Analytics Agenda. *IEEE Computer Graphics and Applications*, 26(February):10–13, 2006.

[44] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma. Frank: a ranking method with fidelity loss. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 383–390. ACM, 2007.

[45] R. Vuillemot and C. Perin. Investigating the Direct Manipulation of Ranking Tables for Time Navigation. *Proceedings of the ACM CHI'15 Conference on Human Factors in Computing Systems*, 1:2703–2706, 2015.

[46] E. Wall, L. M. Blaha, L. Franklin, and A. Endert. Warning, bias may occur: A proposed approach to detecting cognitive bias in interactive visual analytics. *IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2017. To appear.

[47] E. Wall, L. M. Blaha, C. L. Paul, K. Cook, and A. Endert. Four perspectives on human bias in visual analytics. *DECISIVe: Workshop on Dealing with Cognitive Biases in Visualizations*, 2017. To appear.

[48] S. H. Zanakis, A. Solomon, N. Wishart, and S. Dublish. Multi-attribute decision making: A simulation comparison of select methods. *European Journal of Operational Research*, 107(3):507–529, 1998.