Visualizing the Solution Space of Educational Games using TRESTLE

Erik Harpstead
Carnegie Mellon University
harpstead@cmu.edu

Christopher J. MacLellan
Soar Technology, Inc.
chris.maclellan@soartech.com

ABSTRACT: When designing open-ended educational games and other creative instructional environments it is important for designers to understand what learners can do within the space their games afford and whether behaviors across that space are supporting their instructional goals. In this demo I will present several prototype visualization concepts b based on the TRESTLE concept formation algorithm to organize data from student solutions into a tree structure amenable to several kinds of visualization.

Keywords: Visualization, concept formation, alignment, solution space

1 INTRODUCTION

Exploratory data analysis is an important step within the educational data mining process, particularly so in the context of educational games, which generate large amounts player data. Being able to visually inspect trends in data can provide context and perspective on complex statistical analyses and can help guide educational technology design. Unfortunately, hand conducting such analyses on larger data sets, which are regularly generated by educational technology, is often too unwieldy and time consuming to be practical. To overcome this barrier, we developed the Trestle algorithm and an accompanying set of visualizations to help designers and researchers hierarchically organize and explore structured data, such as the kind generated by educational games and other open-ended, creative instructional environments.

Understanding the breadth of approaches that learners can take in these environments and, more importantly, how the game reacts to those approaches, is essential for ensuring effective instruction. In this paper we briefly describe the TRESTLE approach and describe two examples of how it can support the organization, exploration, and interpretation of structured educational game data.

2 TRESTLE

TRESTLE is a concept formation algorithm that incrementally learns conceptual hierarchies given structured examples as training data (MacLellan, Harpstead, Aleven, & Koedinger, 2016). Unlike most learning systems that only support a vector of feature values, TRESTLE supports hierarchical attribute-value lists (represented as Python dictionaries) that contain both nominal (e.g., discrete colors) and numeric (e.g., x and y position) attributes as well as attributes that refer to nested

attribute-value lists, which we refer to as structural attributes (e.g., "block1" might have a nested list that describes its location and color). It also supports relational attributes that can describe relationships between other attributes, such as specifying that "block1" is on top of "block2", e.g., on(block1, block2). The variety of attribute types that TRESTLE can handle makes it broadly applicable to wide range of potential data sets.

Given structured examples described in this representation, TRESTLE can engage in both supervised and unsupervised learning or a combination of the two. Specifically, the system can learn a shared hierarchical organization of both labeled and unlabeled data that enables learning from one kind of data to benefit the other kind. Learning within TRESTLE is incremental, meaning that it is presented with a sequence of examples. Upon receiving each example, TRESTLE sorts each new example into its hierarchy, updating it to reflect the new training data. To guide this learning process TRESTLE uses an objective function called category utility, which is derived from psychological studies of human concept formation (Fisher, 1987). This objective function is similar to the information-gain metric used in decision tree learning, but supports the ability to predict arbitrary attributes of examples.

The hierarchical knowledge structure learned by TRESTLE supports two key capabilities: prediction and clustering. Prediction within TRESTLE operates similarly to learning. The system accepts as input examples with some of their attribute values missing. The system sorts these partial examples into its current organization using the available features and the resulting cluster it is sorted into is used to predict the values of any missing attributes. In this regard, TRESTLE similar to other instance-based learning approaches, such as k-nearest neighbor, but it automatically determines adapts—based on the data—how many examples (the k) to use for prediction. Previous work suggests that TRESTLE's prediction performance is similar to humans on the task of labeling the stability of block structures generated by students in an educational game (MacLellan et al., 2016).

In addition to prediction, TRESTLE can cluster examples both hierarchically and into flat clusters (e.g., into k groups). To generate clusterings, users present TRESTLE with a sequence of complete or partial examples (labeled or unlabeled), which it organizes into a conceptual hierarchy using its learning mechanisms. The resulting hierarchy can be directly returned as a clustering of the data. Additionally, TRESTLE has multiple post-processing routines that can translate these hierarchies into flat clusterings of the examples. For example, it can start at the root of the hierarchy (which contains all examples) and progressively break this top-level clustering into progressively smaller and smaller groups, stopping when it has optimized one of a range of metrics, such as Category Utility, AIC, or BIC. Our analysis of TRESTLE's ability to cluster block structures suggests that it produces groupings of examples that have reasonably high agreement with human-generated clusterings of the same blocks structures (MacLellan et al., 2018), which suggests that TRESTLE might be used to organize large volumes of examples in a way that aligns with how humans might organize the same data.

3 USE CASES

We have designed several visualizations to facilitate interpretation of TRESTLE's outputs, though few have been empirically validated with target users. For example, we built a visualization for exploring the hierarchical clusterings generated by TRESTLE that is powered by D3.js (Bostock, Ogievetsky, & Heer, 2011). Figure 1 shows the this visualization of a hierarchical clustering of examples from the educational game RumbleBlocks (Christel et al., 2012). In this visualization each circle represents a

collection of student solutions to a game level or problem. The leaf concepts (filled in circles) represent specific instances within the dataset while the transparent enclosing circles represent higher-order clusters containing subgroups. The size of each cluster represents how many instances are grouped within that cluster. When a concept or instance is clicked on in the visualization the Attribute-Value Table to the right of the tree shows the distribution of properties within the selected concept/instance for direct inspection of trends in the data. TRESTLE builds probabilistic concepts, so higher-level nodes (concepts) show probabilistic summaries of all the nodes below them in the tree whereas leaf nodes show the attributes of specific instances.

In Figure 1, the nodes are colored based on the outcome of a solution according to the game's log data. Solutions are colored yellow if they are more likely to pass the level, and purple if they are more likely to fail. In this case there is a clear successful cluster (bottom of left branch), and a mostly clear negative cluster (right branch). The outcomes of the other two main branches of the tree (left and right of the larger left branch) are less clear and would potentially warrant further investigation. An analyst can re-color the same visualization according to different properties of solutions to see if there are any correlations in trends that might warrant investigation as design issues.

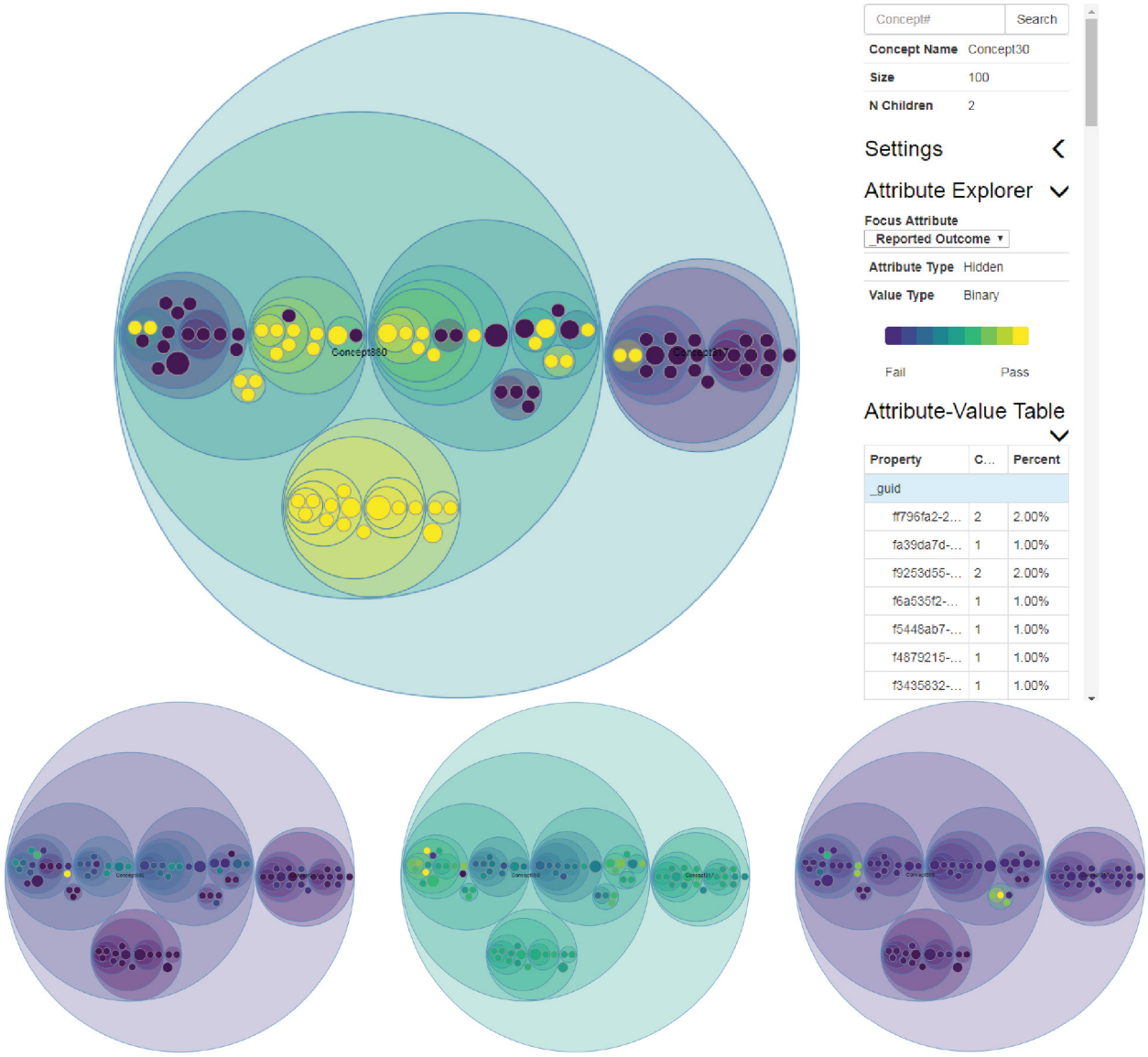

Figure 1. A visualized TRESTLE tree of a sample of 100 solutions to a level in RumbleBlocks. In the top visualization clusters are colored by their likelihood of succeeding on the level (yellow for passing, purple for failing). In the lower three visualizations the same tree is re-colored according to different target properties of game solutions.

Further, the visualization is implemented in interactive D3 code allowing an analyst to click on any of the nodes to inspect in a zoomed in view (as shown in Figure 2). This shift also updates the data in the Attribute-Value table to maintain a grounded sense of the probability distributions represented by the new focused concept.

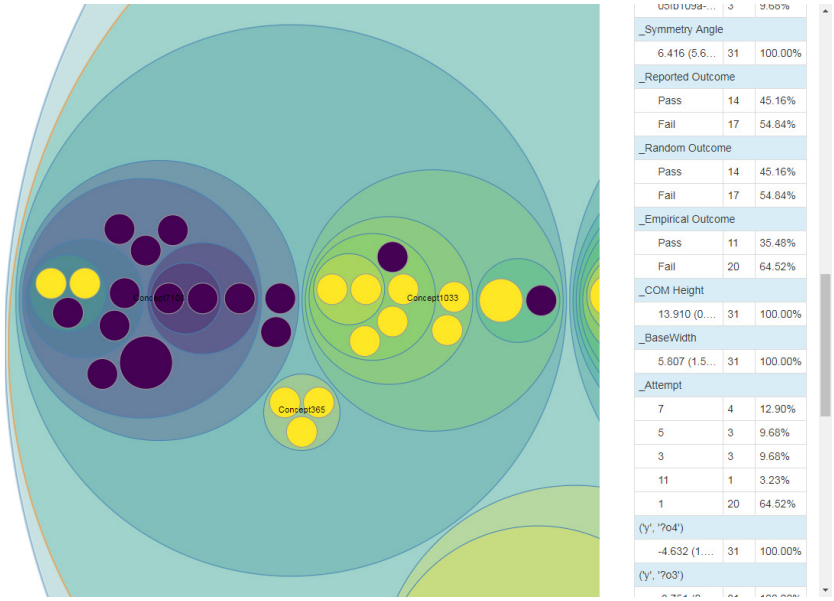

Figure 2. A zoomed in view of a cluster in the TRESTLE hierarchy.

While the hierarchical tree is currently the default output for visualization in TRESTLE, we have also developed more advanced forms of visualization that make use of the TRESTLE data structure in a flattened form. The alignment visualization shown in Figure 3 is based on analyses we have done of the solution space of RumbleBlocks (Harpstead, MacLellan, Aleven, & Myers, 2014). This visualization breaks up the TRESTLE tree into representative clusters that can be plotted according to their properties to look for correlations in data that might be indicative of problems. In the case shown in Figure 3 a principle relevant metric within the game should be predictive of successful performance but the trends across the solutions show that this may not be the case.

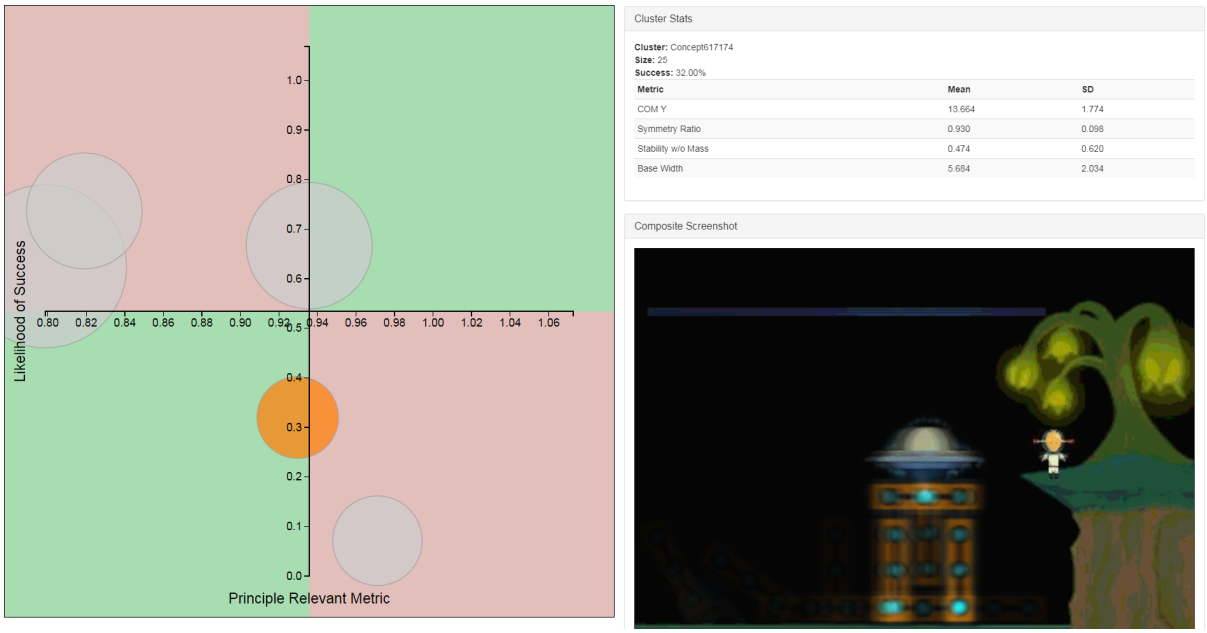

Figure 3. An example of employing TRESTLE to visualize the alignment of clusters of player solutions to a game. A live version of this visualization is available at http://erikharpstead.net/alignment/visualization.html

Digging further into why there may be a misalignment if the game's design, an analyst can make use an instance id attribute that is categorized with each instance to retrieve a corresponding screenshot of the game. These screenshots can be composited together (lower right of Figure 3) to allow an analyst to more quickly examine why a particular trend might be happening within their instructional environment and what actions they may explore to fix the problem.

4 CONCLUSION

Our goal in designing visualizations for TRESTLE is to help analysts to organize their data in way that can support intuitive exploration. We have found this to be particularly useful within datasets from complex instructional environments such as educational games. We hope that others can see utility in this approach and can find a way to apply it to their own contexts.

REFERENCES

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, *17*(12), 2301–2309. https://doi.org/10.1109/TVCG.2011.185

Christel, M. G., Stevens, S. M., Maher, B. S., Brice, S., Champer, M., Jayapalan, L., … Lomas, D. (2012). RumbleBlocks: Teaching science concepts to young children through a unity game. In *Proc CGAMES 2012* (pp. 162–166). IEEE. https://doi.org/10.1109/CGames.2012.6314570

Fisher, D. H. (1987). Knowledge Acquisition Via Incremental Conceptual Clustering Learning ~ Element Performance Element. *Machine Learning*, *2*, 139–172.

Harpstead, E., MacLellan, C. J., Aleven, V., & Myers, B. A. (2014). Using extracted features to inform alignment-driven design ideas in an educational game. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14* (pp. 3329–3338). New York, New York, USA: ACM Press. https://doi.org/10.1145/2556288.2557393

MacLellan, C. J., Harpstead, E., Aleven, V., & Koedinger, K. R. (2016). TRESTLE: A Model of Concept Formation in Structured Domains. *Advances in Cognitive Systems*, *4*, 131–150. Retrieved from http://www.cogsys.org/papers/ACSvol4/paper10.pdf