# A Grammar-based Approach for Modeling User Interactions and Generating Suggestions During the Data Exploration Process

Filip Dabek and Jesus J Caban

**Abstract**—Despite the recent popularity of visual analytics focusing on big data, little is known about how to support users that use visualization techniques to explore multi-dimensional datasets and accomplish specific tasks. Our lack of models that can assist end-users during the data exploration process has made it challenging to learn from the user's interactive and analytical process. The ability to model how a user interacts with a specific visualization technique and what difficulties they face are paramount in supporting individuals with discovering new patterns within their complex datasets. This paper introduces the notion of visualization systems understanding and modeling user interactions with the intent of guiding a user through a task thereby enhancing visual data exploration. The challenges faced and the necessary future steps to take are discussed; and to provide a working example, a grammar-based model is presented that can learn from user interactions, determine the common patterns among a number of subjects using a K-Reversible algorithm, build a set of rules, and apply those rules in the form of suggestions to new users with the goal of guiding them along their visual analytic process. A formal evaluation study with 300 subjects was performed showing that our grammar-based model is effective at capturing the interactive process followed by users and that further research in this area has the potential to positively impact how users interact with a visualization system.

**Index Terms**—Visual Analytics, User Interactions, Analytic Provenance, Machine Learning

## 1 Introduction

As data collection is increasingly streamlined and universal, the process of data exploration, discovery, and analysis becomes significantly more difficult. The popularity of comprehensive datasets have underscored the need for designing new analytical techniques to study large collections of multi-modal measurements and for developing new visual interfaces capable of assisting in the analysis process.

Visual analytics provides a method to address the problem of understanding complex datasets through techniques aimed at presenting data in a way that is easier to understand and navigate. Many users rely on visualization techniques to explore their datasets and ultimately answer their research questions. Unfortunately, users are often faced with visualization techniques and interfaces that require manipulations and filtering that can introduce confusion and result in unwanted distractions or even in information overload. Information overload is the problem that arises when individuals try to analyze a number of variables that surpass the limits of human cognition [18]. In order to evaluate large and complex datasets, most users unconsciously employ dimensionality reduction techniques such as *segmentation* or *conceptual chunking*. *Segmentation* is when a problem is broken into smaller components that do not exceed the human processing capacity and *conceptual chunking* is when different concepts are analyzed independently to reduce the dimensionality of the task [36].

A solution that is able to ensure that users stay on track while exploring their data would address the need to identify and mitigate distractions and information overload while using visual analytics systems, as well as make them more powerful. Unfortunately, in general, little is known about how to develop such a solution that models user interactions and assists users throughout different visualization systems and interfaces. The understanding and modeling of the user interaction process could provide important insights into the development of visualization techniques that are more fitted to the end users and could enable the design of new methods to assist users during the data exploration process.

• *Filip Dabek and Jesus J Caban are with the National Intrepid Center of Excellence at Walter Reed National Military Medical Center in Bethesda, MD. Emails: filip.j.dabek.ctr@mail.mil and jesus.j.caban.civ@mail.mil.*

*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*

In this paper, we introduce the notion of visualization systems understanding and modeling user interactions with the intent of guiding a user to accomplish a task thereby enhancing the visual data exploration process. The challenges faced and the necessary future steps to take in obtaining this goal are also discussed in this paper.

To show the effectiveness of such a solution, we present a grammar-based model that can learn from the way different users interact with the data, determine the most common paths, and automatically provide suggestions to users that, based on given rules, appear to be lost or confused with the data or interface. The technique uses K-Reversible – a grammar induction algorithm – to aggregate a number of subjects' interactions into an automaton, determine a common pattern among $N$ different subjects, build a set of rules, and apply those rules in the form of suggestions to new users with the goal of guiding them along their visual analytic process.

The uniqueness of our approach is that by modeling the user interactions as an automaton and focusing on (1) simplifying the graph, (2) merging identical paths, and (3) handling similar paths we are able to create a model that can automatically generate and provide suggestions. Depending on the size of the automata we were able to obtain between 54.45% and 96.62% compression using our algorithm and show with over 300 users a significant decrease in actions without affecting the users' accuracy.

The rest of the paper is organized as follows. Section 2 discusses some of the related work, Section 3 describes how the user interactions are modeled as a deterministic finite automata, Section 7 discusses our results with a formal user study of over 300 subjects, and Section 8 concludes the paper and discusses some of the necessary future work in this area.

## 2 Background

In the Human Computer Interaction (HCI) community, the mining of interaction data and guiding of users has been well researched: Munro et al. authored a book on social navigation to display a trail of activity to current users [24], Grudin presented the concept of multiple people working as a group in a system towards a task [16], Olson et al. utilized expert constructed grammar regular expressions to identify patterns in interaction behaviors [26], a model where the computer and human both put their best effort into solving a task was presented [1], and Streit et al. presented an expert authored model for visual guidance in a system [35]. Beyond HCI, the field of workflow mining has dealt with extracting information from static logs to identify patterns and insights of system interactions [38, 37]. In the visualization field, Perer and Schneiderman built a system that relied on accurate users to annotate

and share their actions [31], while Willett et al. designed a model for presenting small tips of information to users [40]. We seek to build on this rich literature by removing the requirement of expert constructed models for a specific domain and task.

The capturing and analysis of users' interactions and the use of such information to infer knowledge about the analytical process followed by most users in a visualization has been an active research area [8, 9, 3, 29, 21]. The most widely used method that has been employed to analyze interactions with graphical interfaces has been logging low-level interactions [6, 33, 15]. Low-level interactions are those consisting of mouse clicks and movement. High-level interactions are those that group the user experience into a string representing the sequence of interactions that were taken.

The type of data that is collected to model interactions and analyze different users is key in creating a robust and flexible system. Kadivar et al. captured low-level interactions and allowed users to replay their interactions through a timeline and a series of "undo" and "redo" commands [20]. While historical approaches provide the user with a means to understand their analytic process, users must identify their own missteps and find the correct path on the way to their answer.

Most recently at VAST 2015 Ragan et al. consolidated design knowledge spanning multiple domains into an organizational framework of provenance types and purposes so as to help researchers evaluate designs and mechanisms for capturing provenance information [32]. In addition, several papers focused on capturing and analyzing the user's analytic provenance, such as eye movement data, interaction logs, and thinking aloud videos, allowing for researchers to understand how users arrived at insights in their respective systems [25, 17, 4].

In the past an iconic assistance agent, Clippy, was built by Microsoft with a recommendation algorithm [19] and embedded in the Microsoft Office software aimed at recommending tools and actions that users should utilize. It has been well documented that Clippy was a failure in being polite and providing unobtrusive suggestions to users [39], but it also suffered due to its very short term memory, inability to leverage and learn from many users, and lack of understanding that users can arrive at the same point through a different order of tasks. It is well known that users do require assistance in interfaces from time to time and had Clippy been implemented without the flaws that it possessed, it had the potential to improve both the experience and interface of the Microsoft Office suite through learning of its large scale user base.

Research in the area of understanding the human-machine relationship, as was proposed by Endert et al. [10], has focused on personality traits and how users interact with a system. Brown et al. used low-level interactions, specifically mouse activity, to identify fast and slow users as well as personality traits [5]. While low level approaches may seem promising for identifying characteristics of a user, simple visualizations have been used with low level approaches which allow for low level data to be useful whereas for a more complex visualization the low level data may not prove to perform as well. The need to bridge the transition from low level interactions to high level actions was addressed by Gotz and Zhou in which they analyzed users interacting with a visualization and defined the tiers of the visual analytic process as being Tasks, Sub-Tasks, Actions, and Events [14]. With these developments, it can be concluded that interactions should be captured at a high level rather than a low level because high level actions are application independent and provide a meaningful unit of interaction representation.

With the action tier defined, a visualization system, VisTrails, captured user actions in a visualization as a scientific workflow and analyzed the activity and similarity in branches of actions [3], Xiao et al. used the past visualization states and actions to construct an improved and more relevant initial visualization to display to future users [41], and Ottley et al. found that personality affected the type of search strategy (actions) that a user performs on a visualization [28]. Building on being able to identify a user's interaction method, Gotz and Wen used the premise of actions to analyze users' behavior and cognitive process to suggest alternate visualizations [13].

## 3 Approach

Building on the rich literature in analyzing user's interactions, we identified the need to employ the power of machine learning to aggregate and learn from these interactions to support current and future users in their analytical thought process. As an example, we imagined a system that could capture a large sample of users performing a specific task on a visualization to arrive at an answer to a question, and after utilizing machine learning, the system could assist a new user by identifying which path from their current state would lead them to the answer that they are looking for.

Thus, we outlined our goals as: (i) building a model that is able to represent the interactions that users take to arrive at an insight in a visualization, (ii) utilize machine learning to aggregate and learn the different popular and rare paths that users take in the model, and (iii) build a system that uses the learnt interactions to provide guidance throughout the various interactions of a visualization system. We will describe our process of accomplishing each of these goals in detail below.

## 4 Interaction Model

We model actions that a user takes during their analytical process within a visualization as a finite automaton (FA). FA's are similar to that of directed graphs in that they contain paths, which can be utilized to represent the interactions that $N$ users took, with the ultimate goal of merging similar paths. Before explaining our model further we will give a brief overview of automata.

### 4.1 Automata

Automata are self-operating machines that consist of states and transitions, where the input is compared against the transitions in order to move between states in the machine. Two forms of machines exist in finite automata: *deterministic* (DFA) and *non-deterministic* (NFA). For a given state and input symbol, deterministic machines only have one possible transition, whereas non-deterministic machines can have multiple transitions.

A *prefix tree acceptor* (PTA) is a tree-like DFA built from a learning sample of strings $P = \{p_1, p_2, \ldots, p_n\}$ by converting all of the prefixes $p_i$ in the sample $P$ into states $Q = \{q_1, q_2, \ldots, q_m\}$ and constructing the smallest DFA that is a tree and consistent with the learning sample [7]. For example, given a sample string set $P = \{aa, aba, bba\}$ it can be converted into a set of states $Q = \{q_1, q_2, \ldots, q_7\}$ as illustrated in Figure 1.

One of the basic operations that can be performed on a PTA is a merging operation, which takes two states $(q_i, q_j)$ from an automaton and merges them into a single state [7]. The merging algorithm and pseudocode are shown in Table 1 of Appendix A and an example is shown in Figure 2. The algorithm takes in two states, $q_i$ and $q_j$, that are to be merged together and then takes everything that points into $q_j$ and makes it point into $q_i$. In addition, everything that $q_j$ points to is now made to originate from $q_i$. This removes all of the transitions into and out of $q_j$ and transfers them to $q_i$, allowing the algorithm to now remove $q_j$ from the finite state machine.

Algorithms such as Gold's Algorithm [12], RPNI [27], and K-Reversible [2] start from a PTA and perform operations on it to try and create a DFA that recognizes the target language (the learning sample of strings), which essentially ensures that the DFA is able to process strings similar to the ones provided in building the PTA. By using these algorithms a system can learn a compressed graph containing the probability of taking a given path in a specific visualization state, which can be defined as the number of users that took the given path over the total number that reached the state.

A string $P$ can describe any elements or actions that can be represented as a string. In our system, we chose to represent each node as a state in the visualization and each transition as the high level action taken to reach the next state. Using these high level actions we will be able to model the user's thought process while mitigating issues such as varying screen sizes and resolutions frequently confronted when only collecting low level actions. In addition while explicitly representing visualization states is an unsolved research area and we are unable and do not need to hard-code the visualization state into each node, through the encoding of high level actions into each transition

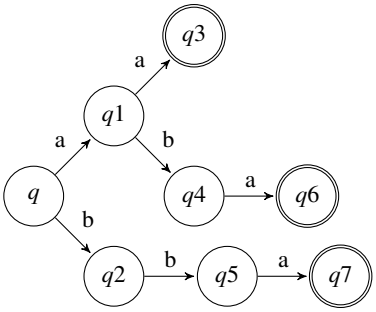


Fig. 1: Example of prefix tree acceptor built from strings $P = \{aa, aba, bba\}$.

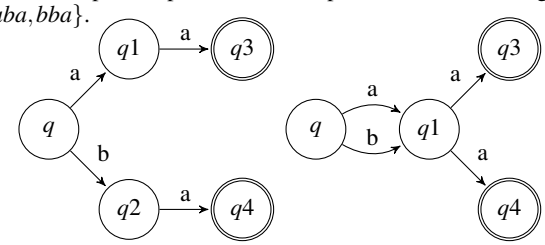


Fig. 2: Example of merging states q1 and q2.

our machine learning approach is able implicitly represent the state of the visualization at each node.

Figure 3 shows an example of the actions that were taken by a single user while answering a given question and using a visualization system. From the figure we can see that this specific user clicked on and off multiple variables until he/she came to a conclusion and could answer the question. More information about this experiment will be discussed in Section 7.

Figure 4 shows an example of the actions that were taken by six individual users while answering a given question. In this example, a sample question was created and subsequently the most optimal way to answer the question was identified, with the optimal way being defined as a path that leads to a correct answer in the fewest number of steps. Note that two users clicked around the interface indicating being lost before following the optimal path and answering the question. By being able to understand the thought process and movement through the visualization, these examples prove that building our model based off of automata are able to effectively capture the paths that users take and that we can now utilize machine learning to learn from the model.

## 5 Learning from Interactions

Looking at Figure 4, there are several goals that need to be achieved for the final graph to be usable for providing suggestions to new users. The goals can be summarized as (1) Simplify the graph, (2) Merge identical paths, and (3) Handle similar paths. While all three of these goals aim to achieve a similar objective, these are three distinct aspects of the graph compression process that need to be addressed.

The first goal, *simplifying the graph*, is necessary because with a large graph it would be more difficult to determine which path lead the user down to an answer; thus a simplified graph will produce more

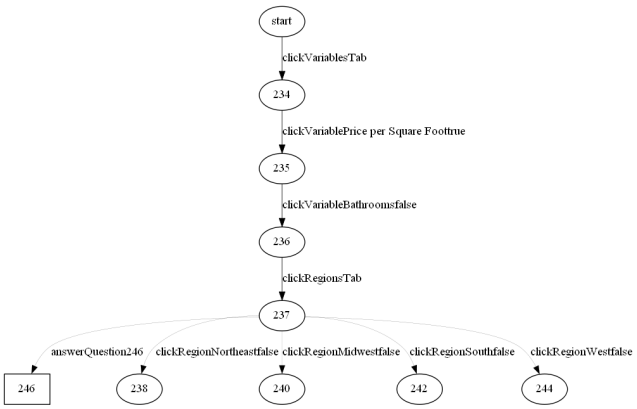


Fig. 3: An example automaton for a user answering question #6 of the user study.

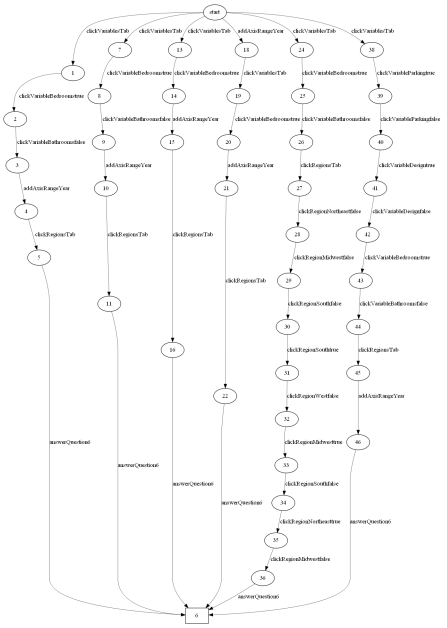


Fig. 4: Automaton of actions generated from six different users while answering a particular question. Note that four users follow a very similar path while two users clicked around the interface many times before following the optimal path that lead to the answer.

effective suggestions to the users. The second goal to *remove duplicate paths* by merging is important to further simplify the graph. Finally, the third goal recognizes that *similar paths* could be considered repetitive and a method to recognize such differences and handle those cases is warranted. This last goal differs from the second goal in that a method to recognize paths that are not exactly identical but similar needs to be identified.

To achieve these goals, and the ultimate goals defined in Section 3, we chose to utilize grammar induction algorithms because these algorithms attempt to merge states in the automata to accomplish the goals of removing duplicate paths and thus simplifying the graph.

The first algorithm taken into consideration was the RPNI algorithm [27, 7], which builds a compact DFA by greedily merging states together. While this algorithm seemed to be promising for this application, it requires positive and negative data to construct the PTA. The positive data consists of a string or a sequence of actions that the final automata needs to be able to accept while the negative data consists of strings that should not be accepted, or in the case of interactions: paths that the user should not follow. In the case of RPNI it generates an automaton that will not accept any of the negative data. For the research realm of visualization, we may not necessarily know if a user answered a question correctly or incorrectly and therefore this notion of positive and negative data does not exist at the the time of creating a graph or in the realm of visualization. Based on these positive and negative states, the RPNI algorithm would not be ideal for this application.

The algorithm that we ultimately chose was the *K-Reversible Grammars Algorithm* [2, 7]. This algorithm, unlike other algorithms, does not factor in negative states resulting in all the states being considered equally. Breaking down the K-Reversible Algorithm, as shown in Table 2 of Appendix A.1, it is classified as a look-ahead language that takes into account a length $k$ sub-sequence of actions at a time. This means that the algorithm starts at a state and looks backwards for up to $k$ states, combining the actions on the backwards path into a sequence. Subsequently, two states can be compared by contrasting their sequences of $k$ length actions. An example of this is shown in Figure 5, where states p and q are similar for $k = 0$ as "a" equals "a", but for $k = 1$ or $k = 2$ they are not similar as "ad" does not equal "ab" and "adc" does not equal "abc".

This K-Reversible algorithm was ultimately chosen for its ability to merge similar states based on a length $k$ sequence of actions as similar sequences of actions can lead to the same state in a visualization.

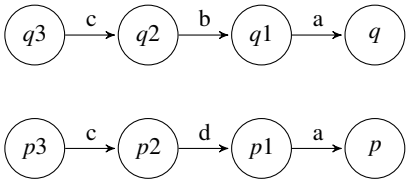


Fig. 5: Example of comparing k actions at a time.

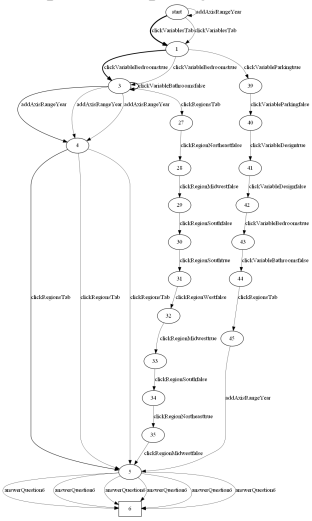

Fig. 6: This figure shows the sample actions of six users after running the K-Reversible algorithm with k=0.

This ability ultimately allows the algorithm to collapse substrings of states that arrive at the same goal, thereby compressing the graph and highlighting the most common paths taken in a visualization system.

The string used for each path in the graph is the action string as defined by the system (e.g. addRange, clickVariable, clickRegion, clickVariablesTab, etc). In addition, at each state of the grammar there exists extra information stored in the state for future use: the amount of time that it took a user to take the action from state A to B, the number of users that took the action, and any other metadata pertaining to the action. For each merge operation performed in the K-Reversible algorithm, this extra information is also merged for each state. This means that the final grammar graph contains the times, probability, and action metadata at each state that can be used in producing suggestions.

Initial tests of running the K-Reversible Grammar against sample data, shown in Figure 4, using the value of k=0 produced the automaton shown in Figure 6, where the bold paths indicate more users having traversed the path. The value of k=0 was chosen for this algorithm as it provides the most aggressive merging, which allows for even the smallest of similarities between users to be identified and merged. In this initial run the algorithm is able to merge the similar paths to reduce the number of possibilities of paths, and thus suggestions, to the user, but there were issues that needed to be addressed with modifications to the algorithm that will be discussed next.

### 5.1 Algorithm Modifications

The K-Reversible Grammar accomplishes the first goal of simplifying the graph, but does not completely solve the second and third goals of removing duplicate paths and identifying and handling similar paths, which is why the modifications are needed. Both of the modifications that will be discussed serve the purpose of reducing the number of paths in the graph. This causes the system, when at a certain state, to have fewer paths to consider for suggestions. Additionally, a merged path will have a higher probability to be taken, as each merge operation increases the number of users and thus the probability of taking that path, compared to multiple individual paths which ultimately leads to more accurate comparisons at each state.

**First Modification** The K-Reversible algorithm strictly considers paths without taking into account that two distinct paths may in fact be the same despite a difference in order of actions. For example, there could be a state $q$ that has paths of [b, a] and [a, b] leading into

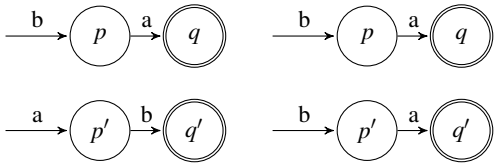


Fig. 7: First Modification

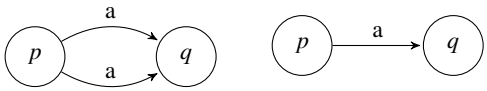


Fig. 8: Second Modification

it, as shown in Figure 7. In these types of occurrences, the ordering of the actions does not matter as both paths lead to the same state of the visualization. This property of the algorithm required a modification, accomplished by reordering the less probable path to be equivalent to the more probable path. Upon reordering and running the algorithm further, it will eventually merge these two paths into one.

**Second Modification** The second modification to the K-Reversible Grammar was to remove identical paths to the same state, that was generally caused by the first modification. During the minimization of the grammar, the algorithm would cause there to be two states, $p$ and $q$, where more than one path went from $p$ to $q$ with the same value as the transition. Therefore, by merging these paths, the grammar would be minimized to the furthest extent. An example of this modification is provided in Figure 8, where the duplicate path of 'a' is merged into one single path.

### 5.2 Classifying Metadata

With the extra action metadata stored at each state, there exists an opportunity to evaluate the action information to further subdivide the actions and thus . For example, at a state that contains the path with an action *addRange* that describes the process of a user selecting a range on an axis, there would be a list of all of the ranges that the users had added at that state to arrive at the next state. As the different ranges that the users selected will affect the visualization in different ways, identifying which "ideology" or thought process that a user belongs to would allow us to alter the action string to reflect this. For example, breaking users' ranges up into two different groups, the action strings could be modified to "addRange1" and "addRange2", resulting in two vastly different ranges not to be merged together or considered similar and thus creating different paths for each group.

The class of algorithms with the best approach to this problem are clustering algorithms, as they allow the various ranges to be put into clusters that are similar. We used the popular and widely known k-means clustering algorithm [23, 11] for clustering the data, but faced the issue of identifying the value of k to use as in this problem k is unknown as all of the ranges selected by the users could have belonged to a single cluster or to multiple clusters. Because of this limitation, we ran the k-means algorithm with varying values of k, one to $\sqrt{(n/2)}$ to be specific as this is commonly used as an upper bound to prevent over-fitting of the value of k, and used the Bayesian Information Criterion [34] to identify the optimal value of k. For our distance metric, we modified the Levenshtein distance function [22] such that rather than using a string and counting the number of character operations, we used an array and transformed each element of the array into a character and then ran the original Levenshtein distance function.

$$D_{a,b}(i,j) = \begin{cases} \max(i,j) & if \min(i,j) = 0 \\ \min \begin{cases} D_{a,b}(i-1,j)+1 \\ D_{a,b}(i,j-1)+1 \\ D_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & otherwise \end{cases} \quad (1)$$

With the modified k-means algorithm and the Levenshtein distance as the distance metric, the K-Reversible algorithm would be run the first time in order to identify the states that required information to be clustered and altered with an id. Following the clustering and alterations, the K-Reversible algorithm would be run a second and final time producing the final graph for providing suggestions.

## 6 Generating and Displaying Suggestions

With the final graph of states and actions constructed by the algorithm described in the previous section, the suggestions could now be generated and the user could be led down the paths of the graph that lead to a final state. Tackling the problem of what suggestion to present to the user, the best suggestion to display was defined as the most probable path given the current state in the graph, and for subsequent suggestions at the same state the next most probable path would be selected. This approach was taken due to the lack of information given if a path leads to a correct or incorrect answer, as the outcome of all answers is unknown.

With the suggestion to display defined, the timing of the suggestions is critical as displaying a suggestion at the wrong time could throw off the user or not displaying a suggestion at the right time could render the approach useless. A basic approach to timing suggestions would be to present the user with a suggestion at each step of the analytical process. However, this constant display of suggestions would remove all of the analytical thinking from the user causing them to blindly follow the suggestions until they had reached the answer in the graph. This blind following of the suggestions would result in no learning being performed or new insights being found. The chosen approach to allow for the user's thought process to not be interrupted, but rather aided, is to provide suggestions at moments where the system predicts that the user is in need of one. This requires a set of rules of when to display and when not to display suggestions.

Firstly, to encourage analytical thinking for the question presented, the suggestions are never displayed immediately at the beginning of a question (at the start state of a graph) as the system does not want to lead the user down a certain path from the beginning, but rather allow the user to explore (the user may discover a new, good path). Beyond the first initial state, there are a set of rules established in the system for when to provide a suggestion to the user, as outlined in Table 1.

Table 1: Suggestion Timing Rules

| |
|---|
| 1. The user is going down the wrong path |
| 2. The user appears to be confused or taking a bad path |
| 3. The user should be on the right path |
| 4. The user is taking longer than expected to make a decision |

*Rules for when to display a suggestion.*

**Rule 1** The first rule of presenting suggestions is identified when the user is going down a wrong path. A wrong path is defined as a path that does not lead to a final state. This can occur when a user takes a path which does not lead them to the answer and causes them to have to revert back to start over. An example of such a path is a user that clicks the incorrect variable name (e.g. clicking "Bathrooms" instead of "Bedrooms") and then needing to backtrack.

**Rule 2** The second rule applies to when a user is exhibiting actions that are characteristic of a user that would take a bad path. This rule can be achieved by identifying the "good" and "bad" paths coming out of the current state; where "good" paths are defined as leading to a final state and "bad" paths are defined as never able to reach a final state. Next the reaction times are split into the "good" and "bad" groups and tested for significance. If the two groups are statistically significant then the user's reaction time for the current state is compared to decide which group he/she belongs to, regardless of which path they actually chose. Statistical significance for this rule is defined by a t-test for two unknown means and standard deviations, which is then compared to a 95% confidence interval. The result is considered statistically significant if the p-value of the confidence interval is less than 0.05. This 95% confidence allows the system to not look for only very strong differences (99% confidence) and to not allow weak differences (90% confidence), therefore creating an approach that is flexible among various interaction graphs.

**Rule 3** The third rule presents a suggestion with random chance to ensure that the user is receiving suggestions and to make sure that they are on the right path. The system starts by setting the random chance to be 1 in X, and for each failed random attempt the denominator (X) is decreased. Once the random generator has succeeded and the suggestion has been displayed, then the random chance is reset back to 1 in X. By default, X is set to the value of 20 because in testing of the system it was found that in a typical interaction sequence, the user performs roughly ten actions and the number 20 allows for the user to receive periodic suggestions, with a 50% chance of receiving a suggestion in a sequence, to make sure that they are on track without overwhelming them.

**Rule 4** The final rule analyzes how long it took most users to make a decision at the current state and waits to display a suggestion if the user has taken longer than the 80th percentile of users. This allows the user to still be able to think about and process what their decision should be, but should they become "stuck" and unable to make a decision this rule will take effect, trying to aid them along.

## 7 Results

### 7.1 Study Design

With a well-defined approach consisting of an algorithm modified to track user interactions and generate suggestions based on rules, we conducted an Institutional Review Board (IRB) approved study (IRB #Y14TO37201) to understand the benefits of our approach in answering research questions. For our user study, the system was loaded with a dataset[1] from the US Census Bureau on the characteristics (# of bathrooms, bedrooms, etc.) of new single-family houses between 1999 and 2012. This dataset was selected for its ability to be universally understood by participants as comparing percentages of houses sold is something that most, if not all, participants can relate to. The dataset is divided into four regions of the country: Northeast, Midwest, South, and West; and further broken down for each house characteristic which consists of: Bathrooms, Bedrooms, Design, Exterior Wall Material, Financing, Foundation, Heating Fuel, Heating System, Lot Size, Parking, Price per Square Foot, Square Footage, and Stories. Therefore, for a given year, the data gives the percentage of houses sold with each type of characteristic. For example, evaluating the Heating Fuel variable the data states that for the year 2003 the Northeast region sold 82% of houses with gas, 15% of houses with oil, and 3% of houses with electricity. Using the visualization system the user is able to browse all of these housing characteristics simultaneously.

Furthermore, the region and characteristic names were obtained directly from the dataset and used in the left sidebar of the system for users to refine the data presented. The user is able to browse and enable multiple regions and characteristics simultaneously allowing for both simple and complex visual data to be explored. On the initial load, the system is configured with all four of the regions and the Bathrooms variable enabled. This approach allows the user to see a basic visualization of data instead of a blank space, encouraging them to analyze the colored lines. Between questions, the interface was reset to the initial state.

To test our approach and reduce the variability from the user's interactions, a single visualization was used within our system. The parallel coordinates visualization was chosen as the visualization technique for this study, despite our approach being visualization agnostic, for its wide range of actions and its complexity, which forces interaction by the user in order to understand the data presented. Figure 9 shows a screenshot of our system. Our framework was designed to be flexible, so there exists an opportunity to study different visualization techniques, however the comparison between individual visualization techniques was determined to be out of the scope of this specific evaluation.

For the suggestions module, the random number for rule #3 defined in Section 6 was set to 20. An example of a user receiving a suggestion is in Figure 9 where the suggestion is colored pink and the user has been informed that they should consider enabling the "Bedrooms" variable as that is the most probable path from their current state in the visualization.

The user study consisted of twelve questions. The twelve free response questions were created by looking for combinations of variables and axis ranges that produced a distinct result (e.g. a single region, variable, or pattern). These questions can be found in Appendix A.2.1. For the purpose of this study, we limited the questions to be

[1] Dataset obtained from http://www.census.gov/construction/chars/sold.html

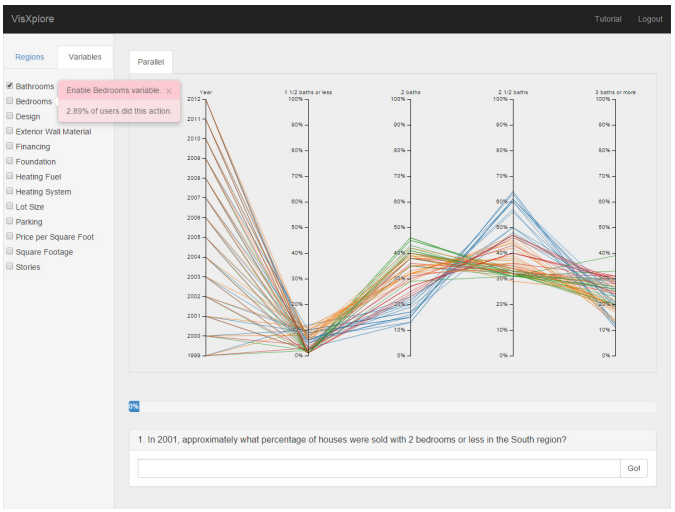


Fig. 9: An example of the visualization system where the regions and variables are separate tabs on the left, checkboxes exist for each region/variable to enable/disable them, the parallel coordinates visualization is in the center, and the question is posed at the bottom along with a progress bar to indicate how far along the user is in the study. In this example, the user received a suggestion (colored in pink) informing them that they should consider enabling the "Bedrooms" variable as users tended to perform that action at their current state in the visualization.

closed-ended rather than open-ended as a system that handles open-ended questions would need to identify the intention of the user which would detract from the objective of analyzing the user's interactions and quest to find an answer. Being able to assist users in closed-ended questions will prove the effectiveness of our approach and thus create an opportunity to expand into more open-ended and abstract questions.

In addition to being closed-ended, the questions were composed with clear answers so as to be able to classify answers as "good" or "bad", but still required the user to put forth effort to interact with the system to discover the answer. An example of this can be found in question 1: "Over all years, which region sold the highest percentage of houses with heat pumps as the heating system?". As mentioned previously the interface begins with the bathrooms variable enabled; therefore the user must enable the heating system variable to be presented with the most pertinent visual data.

In another example, question 9 asks "What region in 2010 had between 40-70% of houses sold with 1 story, and 10% with an attached garage?", which requires the user to find the data for houses with attached garages. Looking at the list of variables, it was found that most users became confused when searching for the applicable variable for attached/detached garages as the variable name provided by the US Census Bureau, "Design", did not make sense to users. The premise behind continuing to use the term "attached garages" and allowing for the confusion in users searching for the "Design" variable was that the rules developed for suggestions should be able to identify the lost users and assist them.

Ten out of twelve of the questions followed the pattern of requiring the user to have to look at specific variables, trying to identify the best region or the trend in data for a certain time period. The remaining two questions, questions 5 and 11, were added as research oriented questions that would be able to show the effectiveness and potential future improvements of suggestions in this context. Compared to the ten questions that require finding a single region through enabling/disabling variables and adding ranges onto axes, the two research questions force the user to go a step further and engage in an analytical thought process where they must look for a pattern between two separate variables in the data. The results of these two questions will help us evaluate our approach towards the

As user interaction data was needed to generate suggestions in the machine learning module, and no data existed at the beginning of the study, the three hundred participants were broken into four groups. The first group consisted of the first fifty participants which received no suggestions, serving as a control group for generating suggestions for future users. The second group, the next fifty participants, received suggestions using data from the first group. The third group, the next one hundred participants, received suggestions using data from the first and second group combined. Finally, the fourth group, the last one hundred participants, received suggestions using data from the previous three groups, but was limited to only the questions where the users had not received any suggestions. With this method, the first group will provide training data for future groups and serve as a base metric for the system as they will not have any influence from suggestions. At the same time, it will be possible to evaluate the amount of data available for generating suggestions versus the effectiveness of the suggestions produced, while keeping the suggestions constant within each group.

Before beginning the study, each user received a tutorial on how to use the visualization system in order to place all users on the same level with respect to visualizations and our system. The tutorial took the user, step by step, throughout the various aspects of the interface and gave them an opportunity at each step to practice interacting and modifying the data and visualization. At the end of the tutorial the user received a sample question that would not be assessed before starting the study. Users had the option to opt out from this sample question. On average users spent 9.07 ($\pm$16.51) minutes going through the tutorial and learning the interface.

While the user completed the study, relevant data was collected and stored in a MySQL database to be used for analysis. The data collected included: the timestamp of when the user was first displayed a question, the timestamp and type of each action taken as well as the state in the interaction graph at the moment the action was taken, the rule that caused a suggestion to appear, the timestamp and result (accepted/rejected/ignored) of each suggestion, and the answers to the questions and feedback survey.

Overall, 300 users over the age of 18 were recruited using Amazon Mechanical Turk. Due to incomplete/invalid submissions (attributed to the nature of Mechanical Turk in which users abandon the study or attempt to provide false answers to quickly finish in a matter of seconds), a total of 430 users completed the study, but only 300 were kept as valid entries. To analyze these results, we will analyze the automata generated followed by the performance, in terms of correctness, number of actions, and confidence of the users.

### 7.2 Interaction Automata

The actions performed by a user were stored in finite state machines for analysis. Furthermore, each users' automata in a group were combined into one single automaton to be used as input to the K-Reversible algorithm as defined in the approach. The input and output of the algorithm will be analyzed through visual inspection and the quality of compression.

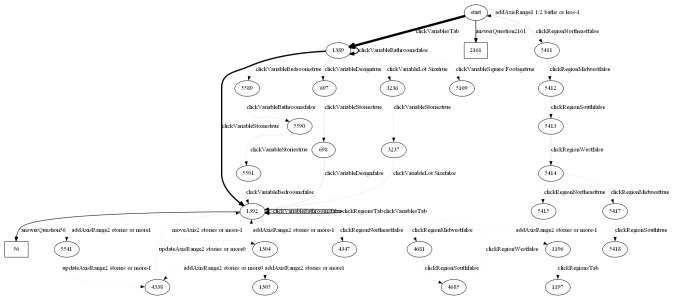

Fig. 10: Automaton, after running the K-Reversible algorithm, of the data for group 1 answering question 7 which is used for generating suggestions for group 2.

Figure 10 shows the automaton, after running the K-Reversible algorithm, of the actions that users in group 1 took to answer question #7. The original automaton, before running through the K-Reversible algorithm, consisted of 197 states and is practically impossible to read, but the compressed consists of only 29 states. Looking at the graph, it

is clear that the paths were successfully merged leaving only distinct paths that the user could take. Furthermore, the most dominant paths that the majority of users took are highlighted with thicker lines and show the user moving down the left side of the graph to the answer marked as a square in the bottom left corner. Inspecting the graph in depth reveals that the bold lines correspond to the user clicking the variables tab, (optionally) turning off the bathrooms variable, turning on the stories variable, (optionally) returning to the regions tab, and then answering the question. This path will clearly assist the user in answering the question as question #7 pertains to the stories variable. In addition, the remaining paths in this automaton provide alternate approaches to answering the question which can be of assistance in two ways: (1) users can be guided away from these paths as they do not lead to the answer that they are seeking or (2) users can be assisted through these paths as they may see something worthwhile that other users were not able to complete and identify. These alternate paths would become especially useful in open-ended research questions/tasks as they represent the less common, but potentially information rich paths.

The automata, by visual inspection, were correctly produced with compressed paths. However, visually the size of the automata grew with the number of users and for some questions was very large due to users needing to interact more. Therefore, analyzing the amount that was compressed through numbers is important as with a large sample of users it is expected for users to follow similar patterns and thus a modest amount of compression indicates that these patterns were identified by the algorithm. In addition a compressed graph that only includes distinct paths, rather than duplicate/similar paths, will aid in generating meaningful suggestions.

| | Percent Compressed | | |
|---|---|---|---|
| **Question** | **Group 2** | **Group 3** | **Group 4** |
| 1 | 86.32% | 74.41% | 96.62% |
| 2 | 80.13% | 87.56% | 89.97% |
| 3 | 62.44% | 84.61% | 93.35% |
| 4 | 69.87% | 82.81% | 81.71% |
| 5 | 84.61% | 92.87% | 90.17% |
| 6 | 86.38% | 94.19% | 93.07% |
| 7 | 85.27% | 89.71% | 90.17% |
| 8 | 80.51% | 88.87% | 91.62% |
| 9 | 64.25% | 90.28% | 91.44% |
| 10 | 79.67% | 89.57% | 89.36% |
| 11 | 54.45% | 69.00% | 77.17% |
| 12 | 72.40% | 94.36% | 92.74% |
| Mean | 75.52($\pm$10.69) | 86.52($\pm$7.81) | 89.78($\pm$5.30) |

Table 2: Average Compression per Question. This table shows the percentage of states that were removed after running the K-Reversible algorithm on the data that was used for each group's corresponding suggestion data.

Analyzing the number of states and paths before and after running the K-Reversible algorithm will show how well the algorithm was able to compress the interaction data. Even though there may be instances where the algorithm is not able to fully compress the data due to a large number of varying paths, this analysis will give a good overview of whether the algorithm is running successfully. For the analysis the automata that were generated for groups 2, 3, and 4's suggestions were used.

The number of states that got reduced on average over all of the automata is 83.94%, indicating that the algorithm is able to compress about half of the paths. Assuming two users taking path A and two other users taking a uniquely different path B, a compression of 50% would be optimal. Therefore, having over 50% compression for varying user interactions paths is representative of a good approach.

For the groups of automata that were generated the average compression for group two was 75.52%, group three was 86.52%, and group four was 89.78%. Based on these compression percentages, it can be seen that the algorithm has a higher compression percentage with a larger amount of users. This indicates that our algorithm is able to correctly identify duplicate paths and merge them, while keeping

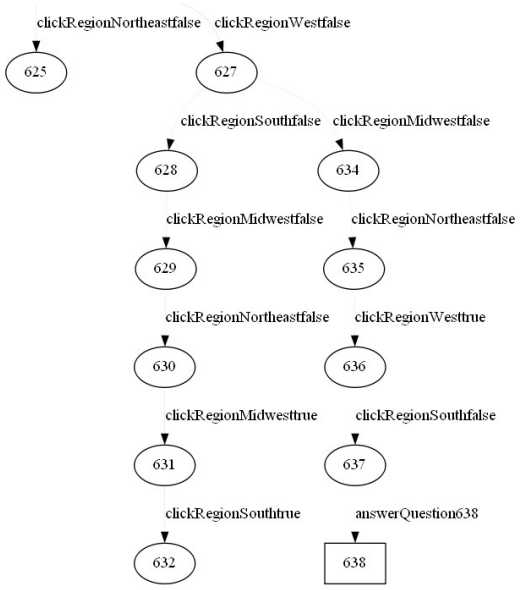


Fig. 11: Part of the automaton for group 1 when answering question 3 where users click between regions.

the unique paths in the automata. However, upon inspecting the automata further there exists an avenue for an overall improvement as some of the necessary states are not merged, primarily when users are clicking between regions as shown in Figure 11.

The compression for each question in the groups of automata ranged from 54.45% to 96.62%. The questions with the highest compression averages varied between groups indicating that there was no singular question that required less actions than another. However, question 11 had the lowest compression average over all groups indicating that users took a wide variety of approaches to answer the difficult research question that we posed. These varying values, as can be seen in Table 2, indicate varying paths taken by users and the algorithm's ability to compress being limited by the number of paths taken by users. The wide range of compression values from 54.45% to 96.62% can be attributed to users performing a wider range of actions for the lower compression questions, as would be expected. In addition, complex questions that had lower accuracies, such as question 4 and 11, were typically the questions with the lowest compression percentages which is consistent with the notion that difficult questions resulted in users performing a wide range of actions. Overall, this data shows us that our algorithm is able to scale and identify duplicate paths well.

### 7.3 Data Analysis

With the knowledge that the automata were generated and compressed correctly and that the visualization context was preserved with the algorithm, analyzing the raw numbers in terms of accuracy, time, and number of actions for users in each question will be important in understanding the impact of suggestions on users in the user study. These results will show how users fared with the questions that were created and if the performance of users, in terms of accuracy, time, and number of actions, was improved.

Of the three hundred users analyzed, the average score was 68.1% ($\pm$0.012), with 3 users earning a score of 0% and 18 users achieving a maximum score of 100%. Out of twelve questions, the lowest accuracy was question 11 with 22% and the sole highest was question 7 with 95% of users answering it correctly, as shown in Figure 12. Combining the data for all questions, the average user took 101.31($\pm$11.196) seconds with a total of 8.23($\pm$0.205) actions to answer a single question. Boxplots of the total time and actions per question can be found in Appendix A. Overall, this data shows that users had a tendency to answer the questions correctly and that the number of actions it took them to reach an answer was consistent with what was expected while constructing the questions.

The two research questions, 5 and 11, produced unexpected results

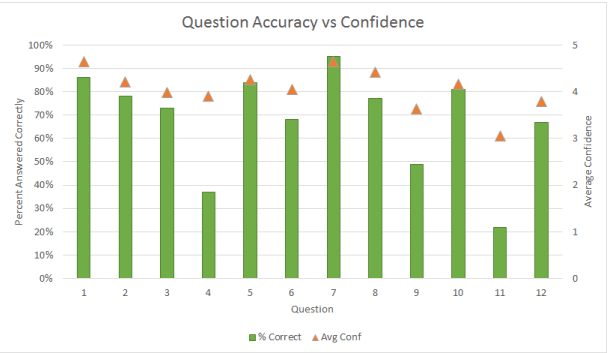


Fig. 12: This figure shows the percent of users that correctly answered each question.

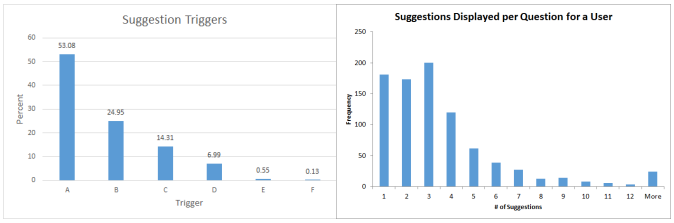


Fig. 13: (left) Triggers for suggestions where A: the user took an amount of time similar to users that took a "bad" path (2415), B: the user is taking longer than normal and is currently unable to make a decision (1135), C: the user took longer than normal to make the previous decision (651), D: random chance (318), E: the user is taking a path that does not lead to an answer (25), F: the user clicked "X" on the suggestion and requires the next one if available (6). (right) A histogram of the number of suggestions displayed to each user for each question.

because while both questions asked the user to analyze similarities and correlation in the data, they were vastly different when answered. Question 5 was answered correctly by 84% of users, while only 22% answered Question 11 correctly. The answers to question 11 show that users were confused by what the question referred to when it asked for "axes" as the majority of the answers contained the the word "regions" or a region from the visualization in the answer. However, the regions determined the coloring of the lines in the parallel coordinates visualization and were not an axis, which indicates that this may have been an oversight by a majority of the users and rephrasing the question could have improved the results. This oversight shows a limitation of suggestions as the only time that this could be recognized is after the user has typed an answer. In addition, there does not exist an action that could be suggested to the user to make in the system to change their thought process to be the "correct" way.

### 7.3.1 Suggestion Rules

To analyze the effectiveness of the suggestion rules, we looked at the two aspects of the suggestions: what triggered the suggestion and what action was taken on the suggestion. Figure 13 (left) shows that the majority of suggestions were triggered from users taking time to make an action that was indicative of users taking a "bad" path, as defined in Section 6. The remaining data shows that a suggestion was rarely ever needed to be triggered for a user taking a path that did not lead to an answer and for when a user clicked the "X" to close a suggestion. This data shows that the system generally made use of all available triggers and that "bad" users were able to be identified.

Results show that the responses by users after being shown a suggestion resulted in: 23.16% of the time users ultimately performed the action that had been suggested to them, 66.24% of the time the user let the suggestion time out and did not follow it, 8.35% of the time users explicitly clicked on the "X" to dismiss the suggestion pop-up, and 0.24% of the time users clicked the "Reject" button to suggestions that altered the interface (such as adding a range or moving the axes). In machine learning, prediction accuracies are compared against random chance to determine the usefulness of an algorithm and in this situation with eight different actions available to suggest and countless variations of each action possible, generating a suggestion that the user agrees with 23.16% of the time is better than random chance. However, this number leaves a lot of room for improvement as the feedback given by users, as will be discussed in Section 7.5 shows that users found the suggestions helpful but did not always need them.

| | W/o Suggest | W/ Suggest | | |
|---|---|---|---|---|
| **Metric** | **Mean(Std)** | **Mean(Std)** | **t** | **p** |
| Accuracy | .734(±.23) | .711(±.26) | -1.154 | .250 |
| Time | 79.58(±44.98) | 76.49(±38.38) | -1.221 | .223 |
| Actions | 7.79(±2.76) | 6.96(±2.24) | -5.098 | .000 |

Table 3: Paired Sample T-Tests on Performance. This table shows the Paired Sample T-Tests results on the various performance metrics accounting for with and without suggestions. The degrees of freedom for each row is 247 and the outliers are removed from the data.

In addition, performing a linear regression on the number of accepted vs rejected suggestions per question for each user reveals that as the number of suggestions displayed increases, the probability of the user rejecting the suggestion also increases. With the linear regression, we found that users were very unlikely to reject the first two suggestions, but the probability greatly increased thereafter. This indicates that as users received more suggestions, they began to ignore them and feel that they were overwhelmed and no longer required them. Therefore, adapting to the user's style of needing more or less suggestions would help from distracting users and displaying needless suggestions, as will be discussed in Section 8 for future work.

When the system was attempting to generate suggestions, the average number of suggestions a user received for a question was 6.53 (±1.27) as shown in Figure 13 (right). Analyzing the histogram shows that the majority of users received a small amount of suggestions (1, 2, 3) which indicates that the suggestion rules could potentially have been too strict and the rules could have been made more relaxed.

### 7.3.2 Performance Gain

When developing the approach, it was important to generate suggestions that would improve the performance of users with suggestions by reducing the time and steps it took to answer questions while increasing the accuracy of the aforementioned answers. Recognizing that users had positive feedback on the suggestions and that the resulting automata were correct provides good insight to the system, but the primary goal of improving users' performance must be evaluated. Performance was evaluated using confidence intervals and t-tests on the number of correct answers and the total time and number of actions to reach an answer.

Due to the number of suggestions potentially impacting the performance of users: a user that receives only one suggestion may not be impacted as much as a user that receives multiple suggestions; we performed t-tests on the data by splitting the data into two groups based on the number of suggestions that were displayed. To determine the points to split the data at, we computed the mean number of suggestions displayed for a particular question and then evaluated the t-tests at the points: mean, mean plus one standard deviation, and mean plus two standard deviations; resulting in multiple t-tests being conducted for each question. Upon performing these tests, we found that questions 1, 8, 10, and 11 were significant at the $p < 0.05$ level in terms of accuracy, questions 2, 5, 6, 7, and 9 were significant in terms of time, and all but one question was significant in terms of the number of actions. These results show that our suggestions were able to significantly improve the users performance in all twelve questions, but an area of improvement exists for future work to identify the optimal number of suggestions to display to a user.

To take an alternate approach, we analyzed the difference in users' performance between questions that they received suggestions versus those that they did not. We used paired sample t-tests to understand if a user performed better under suggestions. For the tests we took the average accuracy, time, and number of actions for questions that a user received suggestions on and those that they did not. We ignored group 1 for this analysis as the users never received suggestions. With this data aggregated, only the number of actions was significant at the $p < 0.05$ level. However, we identified that there existed many outliers in our data where users spent extremely long or short amounts of time on a question causing us to remove these data points from our averages

| Without Suggest | With Suggest | | | |
|---|---|---|---|---|
| Mean(Std) | Mean(Std) | t | df | p |
| 4.13(±0.62) | 4.14(±0.63) | .120 | 247 | .905 |

Table 4: Paired Sample T-Test on Confidence Values. This table shows the Paired Sample T-Test results on the average confidence ratings, with outliers removed, for questions with suggestions versus those without.

by applying a method commonly used in statistics to remove users that were not within two standard deviations of the mean for the particular question. Using this data without outliers we found that the number of actions stayed significant at the $p < 0.05$ level. In addition, for users with suggestions the mean accuracy was higher and the total time was lower. While we were not able to significantly improve accuracy and time we were able to significantly lower the number of actions that a user takes, therefore resulting in one less click needing to be taken to achieve their answer. These results are shown in Table 3.

Analyzing users' feedback, through confidence ratings and surveys, will help determine the usefulness of suggestions as perceived by the users as well as understanding if the suggestions served as a nuisance to the user. It is imperative, even with positive confidence interval and t-test results, that the suggestions are not considered bothersome by the user.

### 7.4 Confidence Rating

At the end of each question, users were asked to select between 1 to 5 stars indicating their confidence in their answer being correct. A value of 1 corresponded to being "absolutely uncertain", while a value of 5 corresponded to being "absolutely certain". Aggregating the average confidence ratings, with the outliers removed, supplied by users for questions with and without suggestions shows that the mean confidence is higher for those that had suggestions. Furthermore, running a paired sample t-test on this data gives a p-value of 0.905, which is not significant. Even though the p-value is not significant, we did not negatively affect the confidence value and the small scale of 1-5 may have impacted our results as there is little room for altering the confidence. A larger confidence scale may have allowed us to see the confidence ratings in more detail.

### 7.5 User Feedback

At the end of the study, the users who received suggestions were asked to provide feedback about the usefulness of the suggestions. The responses to this feedback questions provides insight into the generated suggestions, showing issues where the raw numbers could have been improved through the resolution of issues in the system. The full list of feedback can be found in Appendix A.2.3.

While there was some negative feedback left by users, such as "I thought they were annoying and unhelpful", the majority of feedback consisted of two major ideas: the suggestions were helpful and that the suggestions were distracting and annoying. For instance, two different users clearly summarized both of these by saying "I found them unhelpful and they got in my way" and "Sometimes they were helpful, other times it was annoying because I knew what I wanted to look at". These two ideas show that the majority of the users felt that the suggestions were helpful and that even if users did not change their action our system was able to identify the action that they were about to take. However, on the other hand the suggestions became annoying when the user already knew what they were doing and no longer required them. Therefore, by adapting to the user's learning style and want, or lack of, for suggestions our system could be improved to cause the users to be less annoyed by the suggestions. Users that seem to want more suggestions would receive them more often, whereas users that want less suggestions would receive them when they truly are in need. This could be accomplished by altering the percentiles for the various rules described previously.

Some examples of feedback left in relation to how the suggestions had impacted the users shows that the suggestions were in a large part helpful to the users. Users had said: "they helped reinforce my confidence that I was on the right track," "it helped to correct the path whenever I was going off the track," and "suggestions that appeared during the study was very helpful under critical situation and makes its simple," "I liked them, it reassured me I was on the right path". Overall the optional feedback that users left shows that the suggestions being displayed were consistent with the actions that the user was about to take and helped them find the answer, but that improvements in the timing of suggestions in relation to the user's style (want for suggestions) could net better results.

## 8 Conclusion

In this paper, we have presented the notion of visualization systems understanding and modeling user interactions with the intent of guiding a user to accomplish a task thereby enhancing the visual data exploration process for complex datasets. We have presented our approach, a first step in this direction, for modeling user interactions and automatically generating suggestions in a visualization system. The approach presented uses grammar induction to create an automaton of user actions that is used in the generation of suggestions. Along with the automata, a set of rules were defined for when the system should display a suggestion. The results of the conducted user study show that our approach is effective at modeling interactions of $N$ different users and compelling for generating suggestions that do not have a negative impact on the visual analytic process. With almost all of the users leaving positive feedback, such as that the suggestions helped them "orient", "helped find the answer faster", and "were along the lines with what I planned"; shows that the users were able to be guided along to find the correct answer and accomplish their task. The confidence intervals showed that for 72% of the questions, suggestions reduce variability by producing tighter bound 95% confidence intervals, while the t-test analysis showed that for 91% of the questions, users with suggestions performed the same or significantly better than those without suggestions. This approach of generating suggestions could be applied to a wide range of systems in the field from health care, assisting doctors faced with the challenge of analyzing large, complex, multi-modal datasets, to a workflow system, such as a cash register, assisting users in performing their tasks in less time and steps. This research will help create the ultimate goal of a future contained with systems that assist users in a variety of tasks while attempting to autonomously detect patterns and trends in the interaction and data input to the system.

For the future we look to improve our approach in order to ensure that we are able to build the most optimal support for users. First, based on research shown in the Human Computer Interaction (HCI) field where it has been shown that interfaces need to be developed in a "polite" manner to not detract or distract users [39, 30] we plan to adapt to users' interaction styles so that for users that find suggestions to be distracting the system would slowly reduce the rate at which they are displayed. On the other hand, for users that find them a necessity or that are lost the system would increase the rate. This would ensure that the "Clippy" effect currently present in the system would be removed and the overall user opinion and engagement of suggestions would be improved. Second, we plan to have our system dynamically identify the optimal number of suggestions that a user should be presented with at each question to help maintain better performance consistently. Thirdly, we look to contrast our results against similar, but different visualization techniques such as parallel sets. Finally, we look to our ultimate goal of applying our system to open-ended questions to evaluate the usefulness of suggestions in an environment in which a user session/task does not arrive at a given answer.

As we define our approach to being a first step in this process of understanding, modeling, and guiding user interactions we look for this area of research to be expanded on in order for the positive impact of assisting users with complex datasets in visualizations to be achieved. There exists a lot of work in achieving this goal, but through the (1) exploration of various machine learning algorithms for aggregating users' interactions, (2) improvement in suggestion presentation using existing HCI research, (3) processing of interaction data, such as classifying user undo actions as negative data or identifying various orderings of actions accomplishing a similar task, (4) identifying the optimal time to present suggestions to users as they are completing a task, and (5) choosing the correct path and answer to lead users towards; we see guidance in visual analytics as a promising field with many avenues of exploration necessary.

## REFERENCES


[1] J. Allen, C. I. Guinn, and E. Horvtz. Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications*, 14(5):14–23, 1999.

[2] D. Angluin. Inference of reversible languages. *J. ACM*, 29(3):741–765, July 1982.

[3] L. Bavoil, S. Callahan, P. Crossno, J. Freire, C. Scheidegger, C. Silva, and H. Vo. VisTrails: enabling interactive multiple-view visualizations. In *IEEE Visualization, 2005. VIS 05*, pages 135–142, Oct. 2005.

[4] T. Blascheck, M. John, K. Kurzhals, S. Koch, and T. Ertl. VA2: A Visual Analytics Approach for // Evaluating Visual Analytics Applications. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):61–70, Jan. 2016.

[5] E. 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, Dec. 2014.

[6] P. Cowley, L. Nowell, and J. Scholtz. Glass Box: An Instrumented Infrastructure for Supporting Human Interaction with Information. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005. HICSS '05*, pages 296c–296c, Jan. 2005.

[7] C. de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, New York, NY, USA, 2010.

[8] W. Dou, D. H. Jeong, F. Stukes, W. Ribarsky, H. Lipford, and R. Chang. Recovering Reasoning Processes from User Interactions. *IEEE Computer Graphics and Applications*, 29(3):52–61, May 2009.

[9] M. Eirinaki and M. Vazirgiannis. Web Mining for Web Personalization. *ACM Trans. Internet Technol.*, 3(1):1–27, Feb. 2003.

[10] A. Endert, C. North, R. Chang, and M. Zhou. Toward usable interactive analytics: Coupling cognition and computation. In *KDD 2014 Workshop on Interactive Data Exploration and Analytics (IDEA)*, 2014.

[11] E. W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

[12] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447 – 474, 1967.

[13] D. Gotz and Z. Wen. Behavior-driven visualization recommendation. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, pages 315–324, New York, NY, USA, 2009. ACM.

[14] D. Gotz and M. Zhou. Characterizing users' visual analytic activity for insight provenance. In *IEEE Symposium on Visual Analytics Science and Technology, 2008. VAST '08*, pages 123–130, Oct. 2008.

[15] T. Green, R. Maciejewski, and S. DiPaola. ALIDA: Using machine learning for intent discernment in visual analytics interfaces. In *2010 IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 223–224, Oct. 2010.

[16] I. Grudin. History and focus. *interaction*, 1994.

[17] H. Guo, S. Gomez, C. Ziemkiewicz, and D. Laidlaw. A Case Study Using Visualization Interaction Logs and Insight Metrics to Understand How Analysts Arrive at Insights. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):51–60, Jan. 2016.

[18] G. S. Halford, R. Baker, J. E. McCredden, and J. D. Bain. How many variables can humans process? *Psychological science*, 16(1):70–76, 2005.

[19] E. Horvitz, A. Jacobs, and D. Hovel. Attention-sensitive Alerting. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pages 305–313, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[20] N. Kadivar, V. Chen, D. Dunsmuir, E. Lee, C. Qian, J. Dill, C. Shaw, and R. Woodbury. Capturing and supporting the analysis process. In *IEEE Symposium on Visual Analytics Science and Technology, 2009. VAST 2009*, pages 131–138, Oct. 2009.

[21] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, Dec. 2012.

[22] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady.*, 10(8):707–710, Feb. 1966.

[23] J. B. Macqueen. Some methods for classification and analysis of multivariate observations. In *Procedings of the Fifth Berkeley Symposium on Math, Statistics, and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[24] A. J. Munro, K. Höök, and D. Benyon. *Social navigation of information space*. Springer Science & Business Media, 2012.

[25] P. Nguyen, K. Xu, A. Wheat, B. Wong, S. Attfield, and B. Fields. SensePath: Understanding the Sensemaking Process Through Analytic Provenance. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):41–50, Jan. 2016.

[26] G. M. Olson, J. D. Herbsleb, and H. H. Rueter. Characterizing the sequential structure of interactive behaviors through statistical and grammatical techniques. *Human-Computer Interaction*, 9(4):427–472, 1994.

[27] J. Oncina and P. Garcia. Identifying regular languages in polynomial time. In *Advances in Structural and Syntactic Pattern Recognition, Volume 5 of Series in Machine Perception and Artificial Intelligence*, pages 99–108. World Scientific, 1992.

[28] A. Ottley, H. Yang, and R. Chang. Personality as a predictor of user strategy: How locus of control affects search strategies on tree visualizations. 2015.

[29] S. K. C. P. Pirolli. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. 5(Proceedings of international conference on intelligence analysis.), 2005.

[30] L. Pemberton. Politeness in interaction design. *Romanian Journal of HCI*, pages 1–8, 2011.

[31] A. Perer and B. Shneiderman. Systematic yet flexible discovery: guiding domain experts through exploratory data analysis. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 109–118. ACM, 2008.

[32] E. 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, Jan. 2016.

[33] A. C. Robinson and C. Weaver. *Re-Visualization: Interactive Visualization of the Process of Visual Analysis*. 2006.

[34] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[35] M. Streit, H.-J. Schulz, A. Lex, D. Schmalstieg, and H. Schumann. Model-driven design for the visual analysis of heterogeneous data. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):998–1010, 2012.

[36] K. Tyner. *Development of mental representation: Theories and applications*. Psychology Press, 2013.

[37] W. Van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[38] W. M. Van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. Weijters. Workflow mining: a survey of issues and approaches. *Data & knowledge engineering*, 47(2):237–267, 2003.

[39] B. Whitworth. Polite computing. *Behaviour & Information Technology*, 24(5):353–363, Sept. 2005.

[40] W. Willett, J. Heer, and M. Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1129–1136, 2007.

[41] L. Xiao, J. Gerth, and P. Hanrahan. Enhancing Visual Analysis of Network Traffic Using a Knowledge Representation. In *Visual Analytics Science And Technology, 2006 IEEE Symposium On*, pages 107–114, Oct. 2006.