TwinList: Visualizing List Differences

Leonardo Claudino
Dept. of Computer Science,
University of Maryland
College Park, MD 20742
claudino@cs.umd.edu

Sameh Khamis
Dept. of Computer Science,
University of Maryland
College Park, MD 20742
sameh@cs.umd.edu

Ran Liu
Dept. of Computer Science,
University of Maryland
College Park, MD 20742
ranliu@cs.umd.edu

Ben London
Dept. of Computer Science,
University of Maryland
College Park, MD 20742
blondon@cs.umd.edu

Jay Pujara
Dept. of Computer Science,
University of Maryland
College Park, MD 20742
jay@cs.umd.edu

ABSTRACT
We present a novel visualization tool, TWINLIST, for the purpose of list visualization and matching. Leveraging Adobe's Flex platform and the Prefuse Flare toolbox, we created a rich internet application (RIA) with dynamic animated effects. These animated transitions lead the user through the procedure of matching two lists, using color coding to highlight the similarities and differences between the two. List items can be grouped, sorted and filtered according to their attribute values, to enhance workflow. To illustrate the efficacy of the application, we conduct usability testing for different applications with several domain experts and peers.

Author Keywords
list visualization, medication list reconciliation

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous

INTRODUCTION

A common task in data analysis is comparing list-based information from multiple sources or entities. This problem occurs in everyday life as well as in specialized applications. For example, a consumer interested in making a purchase may wish to compare the features of two products. A newspaper may wish to contrast the rhetoric used in polarized campaign speeches. Or, as a more specialized example, a doctor performing the task of medication reconciliation. In medical reconciliation, a hospital physician must create a treatment plan for a patient prior to discharge. This involves reconciling the medication regime the patient was taking before admission with the medications administered during hospitalization.

A common factor of these applications is that decisions or insights are reached by comparing the constituent items of each list in a structured way. Items in one list may have some similarity or correspondence with items in another, such as specific medications used to treat a particular symptom. The logic necessary to compare these items can be complicated, such as resolving a prescription drug with its generic equivalent. Each item may have attributes that must be taken into consideration when assessing similarity, such as the dosage in the medication list reconciliation task, or term frequency in text comparison. Attributes may also be used to segment list items into categories, such as classes of drugs, or different parts of speech. Finally, list comparison should help reach a conclusion, such as a purchasing decision or a final list of medications. In the latter case, the doctor may wish to "reject" medications that are no longer necessary and "accept" medications that remain in the patient's treatment plan.

Our goal was to design a visualization that allows users to quickly answer the dual questions "What is the same between these two lists?" and "What is different between these two lists?", quickly, accurately and effortlessly. We address these aspects of the list comparison problem in our visualization tool, TWINLIST. Using a spatial organization of data, we provide an intuitive way for users to quickly differentiate between items that are the same and those that differ in two lists of data. We abstract complicated similarity logic by categorizing items as "identical" when they are the same in both lists, "unique" when they appear in only one list, and "similar" when the same item has different attributes in each list. We map these three categories to a workflow and use animation to help users understand each step of the list comparison process. In addition, TWINLISTprovides sorting, filtering, and grouping functionality to allow users to explore different aspects of the data. Finally, the application allows the user to take action by accepting relevant items or rejecting extraneous ones, thus helping users reach a strong conclusion.

Submitted for CMSC 734 Information Visualization.

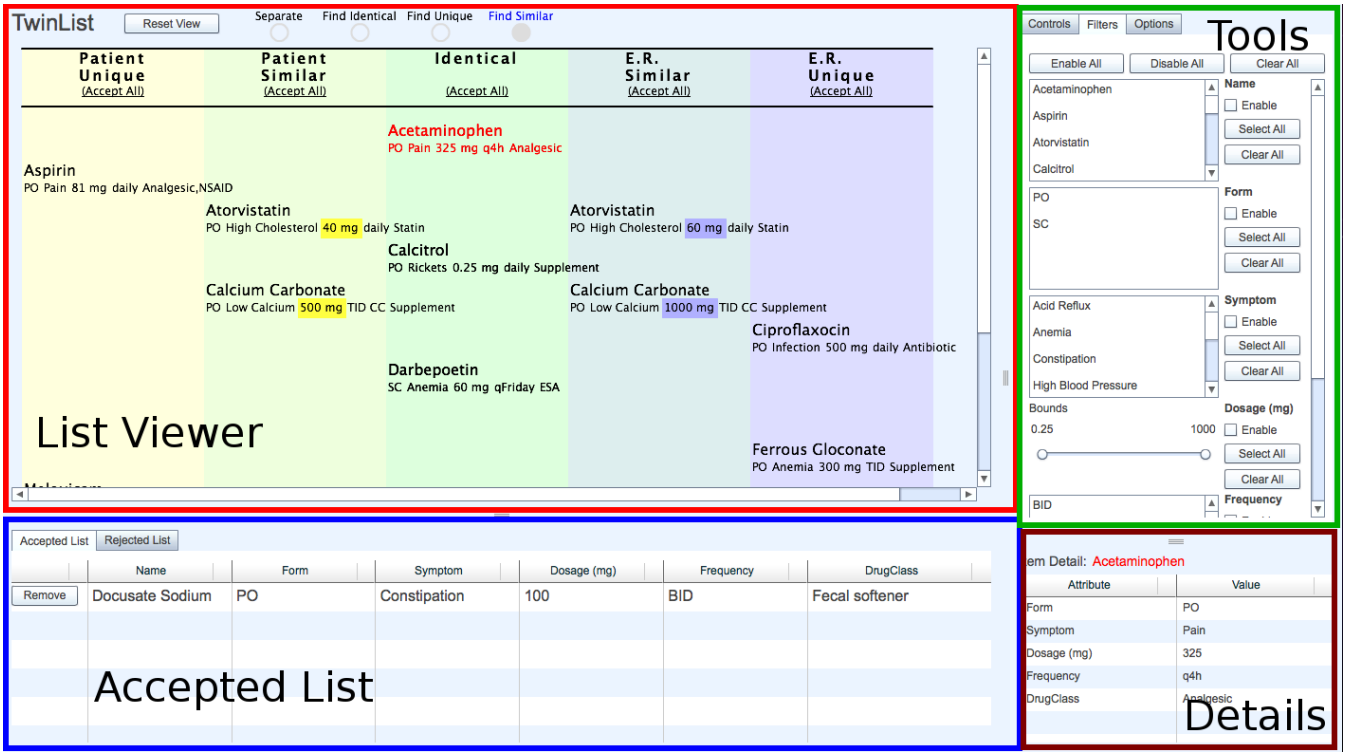

Figure 1. The TWINLIST interface: In this screen capture, the lists have been matched, displaying identical items in the center column, unique items in the outer columns and similar items in between. Color highlighting is used to indicate similar attribute values. In the bottom left panel is the *Accepted* list; in the upper right panel, the *Filter Panel*; and in the lower right, the *Details Panel*, which displays all attributes for the selected item (highlighted in red).

Related Work

We explored multiple existing approaches for the design of TWINLIST, though it seems that the specific problem of list comparison (as we have framed it) has seen relatively little research. Literature in areas such as visualizing text corpora and set membership are tangentially relevant in that they provide a basis for visualizing text similarity. Collins et al. [5] use tag clouds culled from text to compare several lists, using size to indicate frequency and giving a good overview of each document. Subsequent work extended this approach using spatial positioning, making it easier to find unique values [6]. The drawback of these approaches is that they are less useful in comparing specific list items, particularly when attributes must be considered.

Another well-researched area is set comparison. Venn Diagrams have long been used to illustrate intersection and symmetric difference. Kestler et al. [8] propose a modification allowing queries for a specific group of similarities and differences. While we prototyped a solution using Venn Diagrams, they ultimately did not fulfill the needs of our target applications.

Similarity matrices are well-suited to visualizing similarities between one item and multiple others [7]. In addition, we developed a matrix-based visualization that would accommodate several lists. However, we dropped this approach, as the associated benefits were not likely to be relevant to our intended application domains, and it did not address attribute similarities.

Since attributes are of critical importance, we explored approaches that facilitate navigation through list attributes, such as [4] and [11]. Ideas from each were incorporated into our initial designs, but were ultimately dropped as we focused on the core problem of list comparison.

Our decision to use animation was partly motivated by Chevalier et al. [3], in which animation and color are used to highlight the differences in document revisions. This naturally draws the user's gaze to the key areas needing attention. We incorporated these ideas into our design, feeling that animated transitions would make the process of list matching more interactive.

Our final design also incorporates the *Eight Golden Rules of Interface Design* proposed by Shneiderman and Plaisant [13]. Of particular interest were the principles of informative feedback and sequence closure.

Within the medical community, there has been much research [1] [12] highlighting the need for improved medication reconciliation. Preventable Adverse Drug Events (PADEs) are involved in 19% of post-treatment complications. Trials of medication reconciliation policies show dramatic results. In one study, 94% of patients had medication errors, of which a medication reconciliation process eliminated nearly all [12]. Although medication reconciliation has seen such success,

visualization tools for this procedure are still in their infancy. Our discussions with physicians suggested that many do not have adequate tools or a well-specified workflow for medication reconciliation.

IMPLEMENTATION

The first decision in developing an application is the intended form of deployment. This essentially boils down to either a desktop application or a web application. The former offers greater functionality and performance; the latter offers greater cross-platform compatibility and accessibility, since it is available through any internet-enabled computer. Favoring compatibility and accessibility, we chose the latter.

We chose Adobe's Flex framework for rich internet application (RIA) development, primarily because of its animation capabilities and functional, attractive visual components. The decision to use a Flash-based technology does come at some disadvantages: it is not as performant as AJAX (asynchronous JavaScript and XML), it cannot be indexed by search engines and it is currently incompatible with Apple's iOS mobile platform. Nonetheless, we felt that the benefits outweighed these drawbacks.

Our initial prototype used Flex's default DataGrid component to demonstrate the efficacy of the *List Viewer*'s five-column layout. To create the final version, we leveraged the Prefuse Flare toolkit for Flash. The Flare toolkit includes a number of stylish stock visualizations, as well as an extensive API for animation that wraps the Flash core library. Though our visualization did not fit into any of the standard visualizations, we were able to create it using this API.

TWINLIST's data model is entirely data-driven. All names and attributes are automatically derived from the input data. All list data is loaded from XML files, which follow a custom schema. It should be noted that the TWINLIST application does not compute the similarity between items; this is entirely determined by a similarity file (also a custom XML schema), which must exist for every pair of lists being analyzed. The similarity file dictates which items are deemed identical, similar or unique, as well as how similar items differ.

The current version of TWINLISTdoes not support loading user-provided datasets; instead, datasets can be selected from a predefined demonstration set via a drop-down menu in the *Options Panel*. We intend to add the functionality to upload datasets in a future version.

TWINLIST

TWINLIST is a novel list visualization tool for the purpose of illustrating the similarities and differences between two lists. It can be used in many scenarios in which the comparison of two lists of items is required. The primary application that TWINLIST supports is that of medication list reconciliation. However, TWINLIST is equally adept in other applications, such as comparing bag-of-words lists or visualizing gene activity in brain tissue.

TWINLIST's interface consists of four major components: the *List Viewer*, *Tools Dock*, *Accepted/Rejected* lists and *Details Panel*, as shown in (Figure 1). The *List Viewer*, located in the upper left corner of the interface, is where the list matching visualization is performed. The initialized view displays two separated lists loaded from XML files. The *Tools Dock*, in the upper right corner, displays the *Control Panel*, *Options Panel* and *Filter Panel*, which can be made visible one at a time by selecting the appropriate tab. If an item in the *List Viewer* is selected, its detailed information is shown in the *Details Panel*, in the lower right corner. TWINLIST is not merely meant for visualization; it also allows users to take action. List items can be accepted or rejected, whereupon they will appear in the *Accepted/Rejected* lists.

ListViewer

The *List Viewer* is the most important component in TWINLIST's interface. It presents the two lists to be compared, using spatial placement and color to delineate identical, similar and unique items in the lists. Animated transitions are used to enhance understanding of these layout concepts, providing an interactive guide to the list comparison workflow.

The five-column layout

The five-column layout highlights the five possible outcomes of matching two list items. Each item is classified as being either unique to one list or identical/similar to an item in the opposite list. Identical items that occur in both lists are shown only once, and placed in the middle column. The extreme columns (one and five) contain items that are unique to a specific list. Items that are similar to one another are placed concurrently in columns two and four.

Each Item is rendered by its name string, with attributes below. To reduce clutter, the user may wish to select the attributes to render in the *List Viewer*. This can be accomplished via the *Attribute Filter*, located in the *Control Panel*. The attributes that differ between similar items are highlighted to allow users to quickly identify discrepancies.

Color is used to provide a visual cue to the layout. The extreme columns have background colors yellow and slate blue, respectively. The middle column is the combination of these colors, green. Finally, the columns for similar items are yellow-green and blue-green, respectively. The color of the highlighting of similar attributes relates back to the color of each item's associated list. Pale background colors preserve the readability of the text. This palette does not pose problems for most colorblind users. By providing a constant visual cue to the user, we hope to ease the cognitive demand of identifying lists while matching.

To see the *List Viewer*in action, refer to Figure 1. Acetaminophen, Calcitrol and Darbepoetin appear on both lists, so they are placed in the center green column. Atorvistatin appears in both lists. However the dosage of the medication differs between the two lists, with 40mg on the patient list, and 60mg on the E.R. list. As a result, two instances of this item are shown in the second, yellow-green column and the fourth, blue-green column. The conflicting dosages are highlighted

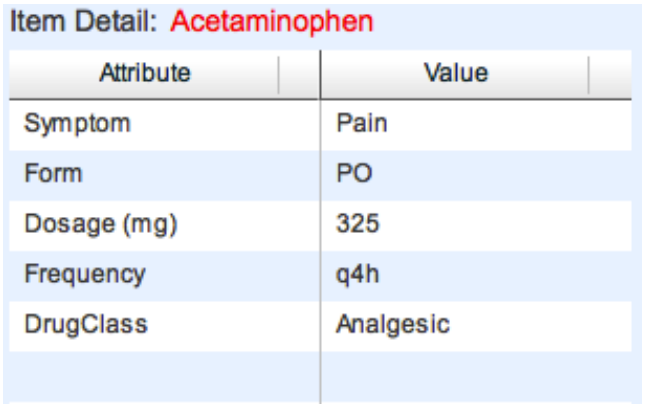

Figure 3. *Details Panel*.

in the attribute text. Aspirin is unique to the patient list, so it is placed in the leftmost, yellow column.

Matching Animation

To emphasize the three stages of list matching, items move to their assigned columns through a sequence of animations, consisting of the following three steps:

1. The *Identical* column header is rendered as the center background color fades in. Following this, identical items move into their respective locations in the center column.
2. The *Unique* column headers and backgrounds are rendered similarly, after which unique items move into the first and last columns.
3. The *Similar* column headers and backgrounds are rendered similarly, after which the similar items move vertically into their respective locations, as the previous animations may have changed the number of rows.

The animation sequence can be viewed from start to finish via the *Match Lists* button, or step-by-step via the animation states at the top of the *List Viewer*, which act as button controls. After leaving the *Separate* state, the *Match Lists* button will become the Reset View button; clicking this will run the animation sequence backwards (from whichever state it happens to be in), thus reseting the visualization. The speed of the animation can be adjusted in relative increments via the *Options Panel*.

Figure 2 shows how the three stages of the list matching animation accentuate the *Identical*, *Unique*, and *Similar* items at each step.

Details on demand

When the user clicks on an item, its information is shown in the *Details Panel*, illustrated in Figure 3. Although the user can choose to render every attribute value in the *List Viewer*, the *Details Panel* provides details-on-demand when the *Attribute Filter* is enabled.

Action taking

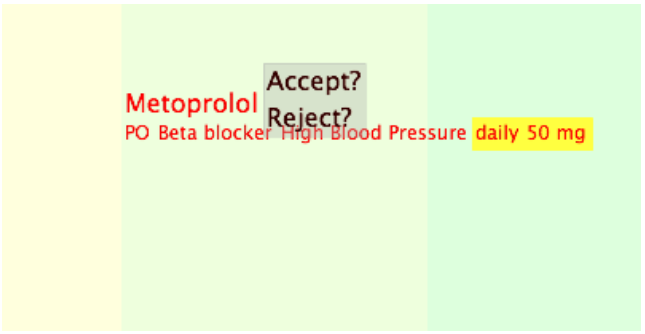

Figure 4. A detail of the action menu.

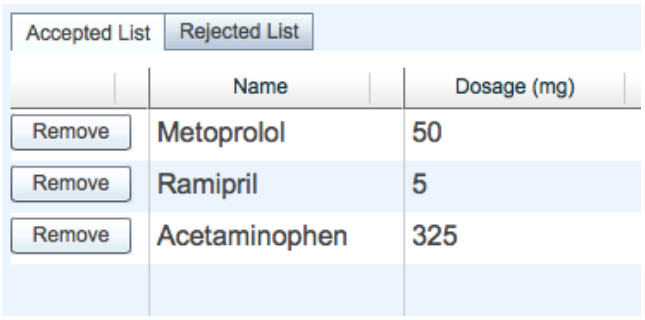

Figure 5. A detail of the *Accepted* list.

After list matching is performed, the user should be able to clearly identify the similarities and differences between the two lists. While in certain applications, analysis alone may be sufficient, others may require the user to take action based on this information. This is especially useful in medication list reconciliation, in which the physician needs to decide which medications will remain in a patient's regime. TWINLIST provides the functionality to accept or reject list items. When the user clicks and holds the left mouse button, a popup menu appears with the option to "Accept" or "Reject" (Figure 4). While continuing to hold the left mouse button, the user positions the mouse over one of these options (which will become highlighted upon mouse-over) and then releases the mouse button to select. (Releasing the mouse button without selecting an option will simply close the popup without any action taken.)

Once the user has selected "Accept" or "Reject", the item is added to the top of either the *Accepted* or *Rejected* lists (Figure 5); the corresponding *List Viewer* item is either removed or grayed-out, according to the user-configurable option (accessed in the *Options Panel*). Actions can be reversed by clicking on the "Remove" button next to each item in the *Accepted/Rejected* lists.

Controls

The *Control Panel*, depicted in Figure 6, contains functionality for organizing and manipulating the the *List Viewer* visualization. This includes the following components:

- *Size By*: scales item text by its value for the given attribute.

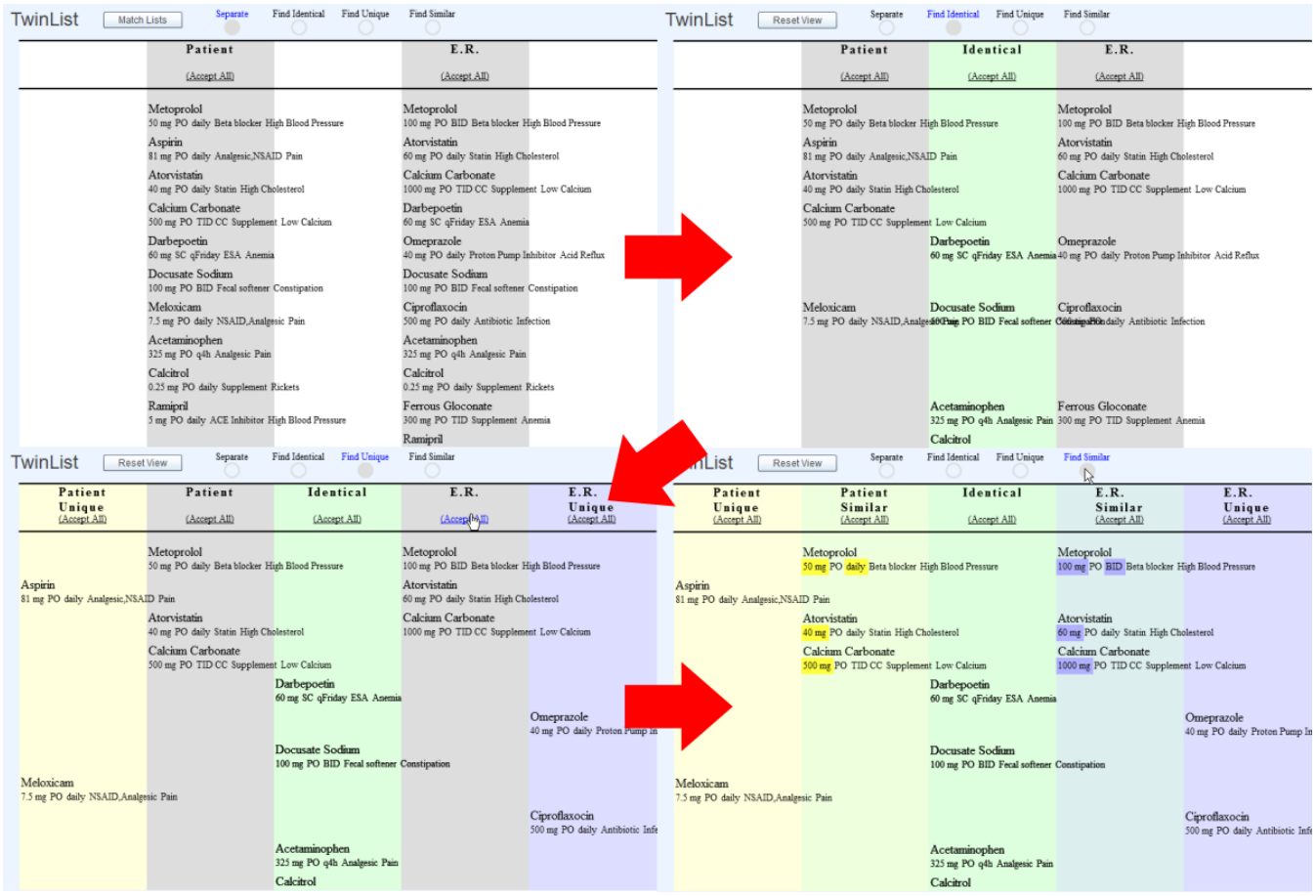

Figure 2. An animation sequence matching two lists of prescription drugs. The original *List Viewer* is shown at the top left. In the top right, the *Identical* items were found. The *Unique* items are found next in the bottom left view. Finally, the *Similar* items are matched with their differences highlighted in the bottom right view.

This control only works with numerical-type attributes.

- *Color By*: colors item text by its value for the given categorical attribute. This control only works with categorical-type attributes.
- *Group By*: organizes items by their values for the given attribute. This control works with either categorical- or general-type attributes (but not numerical).
- *Sort By*: sorts items (within groupings) by their values for the given attribute. This control works with any attribute. Note that the *Group By* control has precedence over this control.

As of the submission of this paper, the *Size By* and *Color By* functionality is not operational, due to time constraints, though these features are planned for future iterations.

For an example of the usefulness of these features, consider the following scenario in the context of analyzing the State of the Union addresses. One might begin by grouping items (words) by their part of speech (assuming part of speech is an attribute). To further organize the display, one might then sort items by name (i.e. alphabetical ordering). The former lets the user analyze syntactical patterns, while the latter enables quick lookup of particular keywords. One can go even further, sizing words by their TF-IDF value, to reveal lexical patterns, similar to a Wordle[14]. In the context of medication list reconciliation, one might wish to group by drug class, size by dosage and color by frequency.

Filters

TwinList's *Filter Panel* were inspired by those of Spotfire[2]. The ability to perform realtime queries and see the changes propagate to the visualization augments data exploration and analysis. For example, in medication list reconciliation, one might wish to display only certain drug classes, or only dosages in excess of a certain amount.[1] Figure 7 depicts a detail of the *Filter Panel*.

Filters come in two classes: categorical and numerical. Categorical filters can be used to display only a specific set of attribute values, taken from a finite set of possible values. Numerical filters can be used to display only attribute values that fall within specified lower and upper bounds. Note

[1]While in practice dosages are typically given on multiple scales — e.g. 325 mg, 30 CCs — we require that all numerical attribute values be normalized to the same scale.

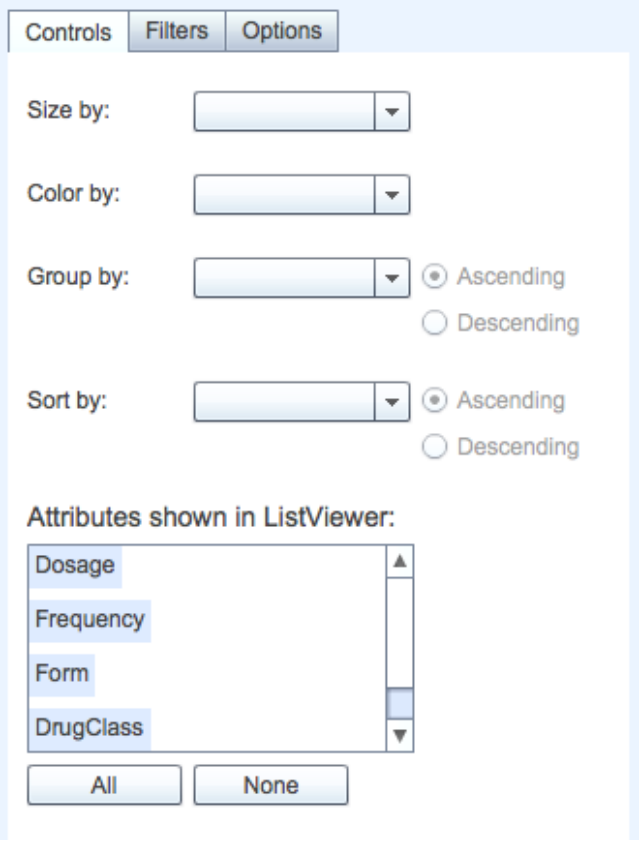

Figure 6. A detail of the *Control Panel*. The *Attribute Filter* (bottom) allows the user to select which attributes to display in the *List Viewer*.

that an item will only display it passes all filters, meaning its attribute values fall within the specified allowable sets or ranges.

A filter can be enabled and disabled via a checkbox displayed to the right of it. Enabling a filter implies that it is active. When a filter is disabled, it has no effect on the *List Viewer*, no matter which values or bounds are set. The *All* and *None* buttons select or deselect all options in a categorical filter, where selection is demarcated by highlighting. A selected value is kept in the visualization. In a numerical filter, *All* simply resets the bounds. The *Reset All* button at the top of the *Filter Panel* resets all filters (though it does not disable them), equivalent to selecting *All* on each.

Options

The *Options Panel* allows the user to control the following global application behavior:

- *Dataset*: selects and loads a dataset from the three demo sets.[2] Upon changing the dataset, the application will be reset, except for the *Options Panel*.
- *Font size*: sets the size of the font used to display item names in the *List Viewer*. Note that the attribute text font size is relative to that of the name, though slightly smaller to reduce clutter.
- *Animation speed*: controls the relative speed of the animations in the *List Viewer*. The speedup is defined as 2^{α}, where α is the slider value. Thus, a value of $\alpha = +1$ will double the speed and a value of $\alpha = -1$ will reduce it by $1/2$.
- *Link identical items*: determines how identical items are treated in the *List Viewer*. When linked (the default), highlighting affects both items and accepting/rejecting either will hide the other from the *List Viewer*. When unlinked, they are considered completely separate items.[3]
- *After item accept/reject*: determines the behavior of the *List Viewer* after accepting or rejecting an item. *Gray out* will simply gray the text and disable mouse actions; *Remove* will hide the item from the *List Viewer*.

[2]In future versions, the user will be allowed to upload a dataset using our proprietary XML schema.

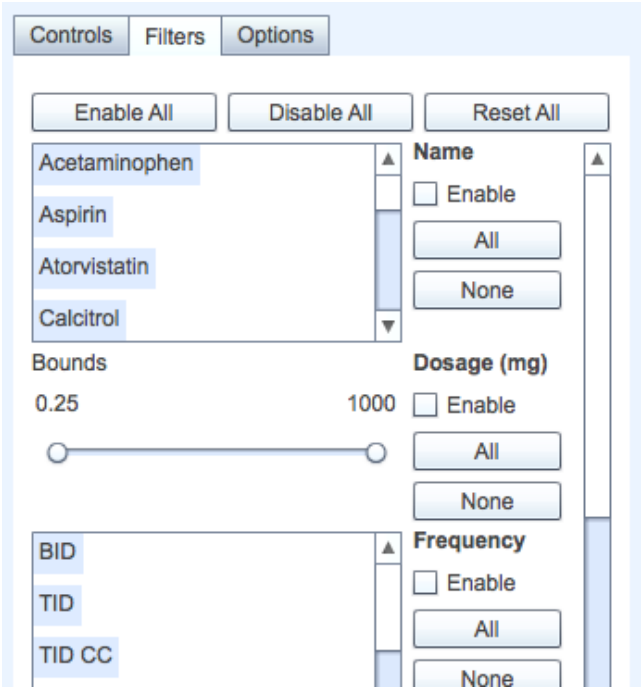

Figure 7. A detail of the *Filter Panel*.

EVALUATION

Our goal was to deliver a working prototype, not a finished, fully-functional application. As such, we were more concerned with qualitative rather than quantitative feedback on the features of our visualization system at this point. According to the categorization proposed by Lam et al [9], the evaluation scenario best matching our present purposes is the *User Experience (UE)* test case. The goal of the following experiments was to observe how users reacted to the visualization and interactive features offered by TWINLIST. We will use their feedback to improve the system's design in future versions.

[3]Unlinking identical items implies that both items can be accepted or rejected, independent of the other.

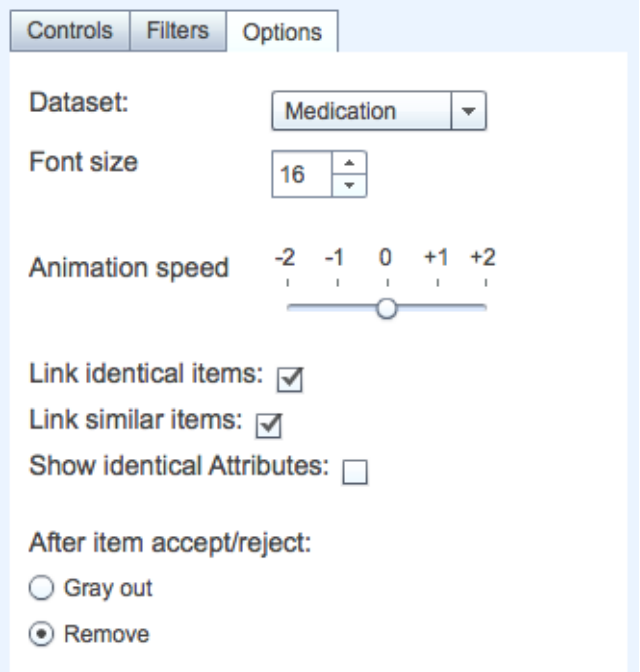

Figure 8. A detail of the *Options Panel*.

Experiment Design

Test subjects came from two populations: two physicians and two graduate students in Computer Science (CS), both in the field of Human-Computer Interaction (HCI). The former used TWINLIST to perform medication reconciliation. The latter was tested on the car data set (lists of features of two cars, one list per car). The testing protocol proceeded as follows: first, the user watched a 7-minute video demonstrating basic usage. Following this, the user was allowed to ask questions. Next, the user interacted with the system unaided, performing either specific tasks (medication reconciliation case studies) or general, free exploration ones (car data case studies). While using the system, subjects were encouraged to *think aloud* [10]. The entire experiment was recorded in both audio and screen capture video.

Case 1: Med. reconciliation, A. Z. H., male, 34 y. o.

Overall, the participant performed the tasks as expected and reported anticipated results. A. Z. H. purported to be very comfortable with computers. He does not perform medication reconciliation on a regular basis, except when doing inpatient aid and discharge from the hospital, for which he does not use any software. This is how he describes his TWINLIST experience: *"It's really impressive, the way you organize things is great. It's definitely a step in the right direction . . . "*. He was very enthusiastic and provided several suggestions, most of all worth including in this section.

Results

The following notes summarize A. Z. H.'s experience:

- There is need for an indication that there are more items than what can be currently seen in a column. The scroll bar alone is insufficient.
- It is difficult to get to the *Accept/Reject* pop-up menu when accepting/rejecting an item. He had trouble with the click-and-hold behavior, wanting instead to use the mouse's right-click button.
- To send items from *Accepted/Rejected* lists back to the viewer one-by-one requires quite some effort when you have many items in those lists. A *Remove All* button would be desirable.
- Warning dialogs or undo may be needed when accepting and rejecting items, since it is a delicate operation in medication reconciliation.
- In the control panel, drop-down boxes should indicate what the default *Group By*/*Sort By* attributes are, instead of blank captions. (The default is actually the original ordering.)
- It would be desirable to shorten vertical empty spaces in between items to minimize the need of scrolling.
- The number of visible item attributes should be shortened in the *List Viewer*, since reconciliation is performed, in practice, based on only a few (e. g. the *Indication* feature).
- For items in the *Similar* columns, the item pop-up menu should have a third option, to automatically reject an item of a list if its counterpart at the other list was accepted. According to the participant, this would be the likely outcome in almost every such context.
- The participant suggested that we should change the way items are displayed so that, for each list, you have items sorted by some *uniqueness level*; for example, by vertically ranging from identical to unique. This would add more meaning to the vertical structure of the list, which is somewhat lost after lists are matched, but would represent a significant change in the concept.
- The participant felt that placing the *Accepted/Rejected* lists at the bottom of the interface may be suboptimal for tracking what happened to processed items, especially when the user gets distracted.
- The participant missed being able to interact on the spot with accepted/rejected items that are grayed out. In his words: *"If it is grayed out, what if (. . .) I really want to do the 100 mg, is there any way to click and have it re-instate (. . .) or un-reject/un-accept"*. He believes that if he could interact with grayed-out items on the viewer, the *Accepted/Rejected* lists might not even be necessary, thus allowing more room for the main visualization. As such, the list of rejected/accepted items could be output as the final product of the interactive process and would be activated by pressing a button. As he suggested: *". . . Here is your final list of medications, this is your rejected list over here, and when one finally accepts, export to (. . .) or put in patient's medical record . . . "*.
- The participant suggested an additional *Sort By* criteria: in his own words, *"medications that haven't been dealt with*

yet." We translate this as a grouping of medications based on their accepted/rejected status. This is a useful, general concept applicable to other list matching problems.

- The participant mentioned the importance of filters, especially to filter by route (e.g. I.V. medications).

Case 2: Med. reconciliation, M. G., male, 62 y. o.
Unlike the previous participant, M. G. had more trouble performing his tasks. His reactions clearly indicated that he expected the system's terminology to be geared more towards medication list reconciliation than general list matching. As the matching animation began, he expressed feeling overwhelmed: *"Wow, what just happened there?"*. M. G. considers computers somewhat hard to use (he reported a difficulty score of 4 out of 5). This subject performs medication reconciliation every day.

Results
The following notes summarize M. G.'s experience:

- In his opinion, one item's list of attributes should not exceed the width of its column. He felt that the piece of the attribute list that spills into the next column could be associated with another item in that column, which could lead to a serious confusion: as he pointed out, the dosage of one medication could be mistakenly associated with another item.
- The participant became confused by the step-wise animation buttons when told to accept all identical items. When asked to accept all identical items, he clicked *Find Identical* instead of *Accept All*. The animation then proceeded half-way through and he realized his mistake. He tried to fix it and ended up clicking on *Find Unique*. Finally, when he tried to accept/reject items, he realized that similar items across both lists were not in the same row of the viewer. The reason was that he had not completed the list matching animation. This indicates that the list matching should be performed as an atomic operation, instead of allowing the user to step through the stages. (This is perhaps supported by the fact that the first physician never even touched the step-wise matching feature.)
- An unexpected outcome of this test was that the participant somewhat associated temporal order from left to right, as if the first list represented an event that necessarily came before the second list. He compared the dosage of the drugs at the Patient's *Similar* column (left) with the corresponding drugs/dosages at the E.R. column (right), and tried to infer some sort of temporal pattern regarding how the patient was dealt with in the E.R.: *"Conceptually, I think it is a nice setup, actually, because you can try to deduce what went on during the stay in the E. R. For these drugs, they decided that the patient needed more of that, because they increased the dose ..."*. Although this was not the intent of the synthetic data, nor the visualization, to associate temporal meaning to the visualization might be an interesting path to explore in the future.
- The issue of having to unnecessarily reject a similar item after accepting its counterpart appeared again, particularly if dosage is the single attribute that changed across items: *"If it is absolutely the same in every respect, except for the dosage amounts, you can't give two dosage amounts simultaneously ..."*. (Note: this feature has been added in the most recent version.)

Cases 3 and 4: Car data, T. L., male / A. N. S., female.
Both participants are doctoral student in CS/HCI. In particular, the first one has a background in interactive design and is very familiar with usability testing. T. L. had trouble to grasp the list matching idea at first, but after some explanation and background information he understood the concept well. He had very few problems while interacting with TWINLIST, and was particularly impressed with the animation. A. N. S. performed the experiment very smoothly and had no problem interpreting the visualization. She immediately came up with a plan on how to find similar items and detect differences, clearly describing the steps she took to reach her results. A minor difficulty she encountered was to operate the *Accept/Reject* pop-up menu. In fact, it is a recurring problem, since it also happened to the first participant in the medication list reconciliation experiment. However, after A. N. S. was reminded on how to operate the feature, she was quickly able to make it work.

Results
The most prominent insights from these experiments were:

- T. L. complained about the size of the *List Viewer*. This brings back some suggestions of participant A. Z. H., who pointed to the lack of "vertical meaning" after matching is performed. In fact, when items are removed from the viewer, the panel is left with apparently unnecessary, empty rows, potentially increasing the need of vertical scrolling.
- To perform list matching, T. L. seemed to prefer the step-wise matching buttons to the *Match Lists* button on the left. We believe this was mostly due to the exploratory nature of this particular experiment, that the participant was encouraged to explore the interface as thoroughly as possible.
- A. N. S. was very impressed with the filter options but did not like the fact that the viewer changes size after filtering. This is a side-effect of the automatic sizing that could be removed in future iterations.

CONCLUSIONS AND FURTHER DIRECTIONS

Our goal was to create a list comparison tool to assist list matching for many applications, with a particular focus on medication list reconciliation. We designed an intuitive spacial organization for differentiating identical and unique items, as well as highlighting the differences in similar items. The use of animation reinforces the user's understanding of this layout and emphasizes the procedural aspect of list matching. The included filtering, grouping and sorting tools allow the user to thoroughly explore a dataset.

By conducting a number of case studies, we gained additional insights about the design and use of this tool. Evaluation results revealed that testing subjects from both popula-

tions were positively impressed with the tool and supported our expectations. From the medication list reconciliation results, we can say that both users approved of our five-column layout and color-coded highlighting (attribute-based differences), two key concepts behind TWINLIST. We also learned that the step-wise list matching feature appears to be superfluous and potentially troubling, suggesting that list matching, both from the procedural and visual perspective, should be performed as an atomic sequence. The exploratory experiments performed with CS/HCI students revealed some flaws in our interface designs, from use of vertical space to difficulties with the input controls. All of these comments will be taken into consideration as development moves forward on TWINLIST.

ADDITIONAL AUTHORS

REFERENCES

1. Joint commission on accreditation of healthcare oganizations: Using medication reconciliation to prevent errors. *Sentinel Event Alert*, pages 1–4, January 2006.

2. C. Ahlberg. Spotfire: an information exploration environment. *SIGMOD Rec.*, 25:25–29, December 1996.

3. F. Chevalier, P. Dragicevic, A. Bezerianos, and J.-D. Fekete. Using text animated transitions to support navigation in document histories. In *Computer Human Interaction*, pages 683–692, 2010.

4. R. Chimera. Value bars: an information visualization and navigation tool for multi-attribute listings. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '92, pages 293–294, New York, NY, USA, 1992. ACM.

5. C. Collins, F. Viegas, and M. Wattenberg. Parallel tag clouds to explore and analyze faceted text corpora. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, pages 91–98. IEEE, 2009.

6. D. Conway. Building a better word cloud. `http://www.drewconway.com/zia/?p=2624`.

7. B. Cornelissen and L. Moonen. Visualizing similarities in execution traces. In *Proceedings of the 3rd Workshop on Program Comprehension through Dynamic Analysis (PCODA)*, pages 6–10, 2007.

8. H. Kestler, A. Müller, T. Gress, and M. Buchholz. Generalized Venn diagrams: a new method of visualizing complex genetic set relations. *Bioinformatics*, 21(8):1592, 2005.

9. H. Lan, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale. Seven guiding scenarios for information visualization evaluation. Technical Report No. 2011-992-04, Department of Computer Science University of Calgary, 2011. `http://www.lri.fr/~isenberg/publications/papers/Lam_2011_SGS.pdf`.

10. C. H. Lewis. Using the "thinking aloud" method in cognitive interface design. Technical Report No. RC-9265, IBM, 1982.

11. T. Masui. Lensbar - visualization for browsing and filtering large lists of data. In *In Proceedings of InfoVis'98*, pages 113–120. IEEE, 1998.

12. P. Pronovost, D. B. Hobson, K. Earsing, E. S. Lins, M. L. Rinke, K. Emery, S. M. Berenholtz, P. A. Lipsett, and T. Dorman. A practical tool to reduce medication errors during patient transfer from an intensive care unit. *Journal of Clinical Outcomes Management*, 11(1):26, 29–33 (14 ref), 2003.

13. B. Shneiderman and C. Plaisant. *Designing the User Interfaces*. Pearson Addison-Wesley, 2005.

14. F. B. Viegas, M. Wattenberg, and J. Feinberg. Participatory visualization with wordle. *IEEE Transactions on Visualization and Computer Graphics*, 15:1137–1144, November 2009.