Capturing and Supporting the Analysis Process

Nazanin Kadivar, Victor Chen, Dustin Dunsmuir, Eric Lee, Cheryl Qian, John Dill, Christopher Shaw, Robert Woodbury

School of Interactive Arts and Technology

Simon Fraser University

ABSTRACT

Visual analytics tools provide powerful visual representations in order to support the sense-making process. In this process, analysts typically iterate through sequences of steps many times, varying parameters each time. Few visual analytics tools support this process well, nor do they provide support for visualizing and understanding the analysis process itself. To help analysts understand, explore, reference, and reuse their analysis process, we present a visual analytics system named CzSaw (See-Saw) that provides an editable and re-playable history navigation channel in addition to multiple visual representations of document collections and the entities within them (in a manner inspired by Jigsaw [24]). Conventional history navigation tools range from basic undo and redo to branching timelines of user actions. In CzSaw's approach to this, first, user interactions are translated into a script language that drives the underlying scripting-driven propagation system. The latter allows analysts to edit analysis steps, and ultimately to program them. Second, on this base, we build both a history view showing progress and alternative paths, and a dependency graph showing the underlying logic of the analysis and dependency relations among the results of each step. These tools result in a visual model of the sense-making process, providing a way for analysts to visualize their analysis process, to reinterpret the problem, explore alternative paths, extract analysis patterns from existing history, and reuse them with other related analyses.

INDEX TERMS: I.3.8 [Computer Graphics]: Applications-Visual Analytics, I.6.9 [Visualization]: information visualization, H.5.2 [Information Interfaces & Presentations]: User Interfaces - Graphical User Interfaces (GUI)

KEYWORDS: Visual Analytics, Sense-making, Analysis Process, Visual History

1 INTRODUCTION

In the investigation of large collections of text documents, an analyst must often make connections between disparate bodies of evidence in order to learn about observations that have been made, and to form hypotheses about how these observations are linked. An analyst will want to discover unknown information embedded in the document corpus, investigate different evidentiary trails, weigh their quality, and compare their strengths and weaknesses. For example, an intelligence analyst may analyze field report documents, or a computer scientist may investigate reports written about a software library.

For situations where the document collection is small, the question is tightly constrained, and the period of investigation is short, an investigator can easily develop sufficient knowledge of the domain to come to a high-quality answer to the question at hand. With large document collections, open-ended questions or long periods of investigation, management of the document corpus, the hypotheses formed, and the avenues investigated become much more problematic. The complexity of the analysis process itself must be managed so that the analyst can review former analytical steps, explore new avenues, and maintain a record of what prior analysis paths succeeded or failed to generate useful knowledge. As new data arrives, the analyst will want to update the state of the current analysis, rerunning previous queries and validating previously drawn conclusions.

Significant research has been conducted on improving the sense-making process by providing more convenient visualization techniques [7, 13]. Most of these efforts focus on visualizing datasets to more easily reveal the narratives within. There also exists a growing body of research on capturing and understanding the analysis process [22]. So far there has been relatively little effort focused on improving the analyst's awareness and understanding of the analysis cycles by allowing them to easily review their analysis interactions.

In this paper, we introduce CzSaw, a visual analytics tool that captures and visualizes the analysis process and history of user interactions with the data. CzSaw includes a number of visual data representations that allow analysts to explore and understand the data. During the data exploration process, all of the interactions with these data representations are both captured by CzSaw in a scripting language and visualized in a visual history view so that analysts can look at the history of their interactions to find repetitive patterns and identify alternative investigative avenues. These other directions can then be explored without losing track of previous work. It also provides the ability to explicitly program the process, so that the end result conveys a chain of analytic logic. Steps in the analysis can be replayed on new data or in variations to remove the burden of repetitive actions.

Providing the visual history benefits analysts in several ways. First, the analysis process is usually meticulous and tedious. During this process, the analyst may get distracted by other issues and have to pause and resume again later. With a visual channel of analysis history, we can take advantage of human visual memory to help the analysts recall the process after a hiatus [26]. Second, the analysis process involves exploring multiple possibilities. Our visual representations of the history help analysts to see these alternative possibilities and take steps back to change the state of the system to a previous state which can also be useful for correcting errors. At the same time, annotation on a visual representation of the views is more effective than making

• *E-Mail: {nka23, yvchen, dtd, ela10, cherylq, dill, shaw, rw}@sfu.ca*

notes outside of the system. The importance of note taking within the system is seen by its inclusion in many recent visual analytics programs [10, 14, 15, 24].

Another way CzSaw helps analysts understand their analysis process is by maintaining a dependency graph of the data that the analyst has explored and made linkages between. A common example of a dependency graph is the spreadsheet, where a numerical change in a cell propagates updates in all formulas that refer to that cell. In CzSaw, any result set of data generated in the analysis process is represented in the script as a variable and in the dependency graph as a data object. Changes in an "upstream" data object propagate updates to data objects that depend on it (located "downstream" from it in the dependency graph). CzSaw presents a visualization of the dependency graph. Using the script, analysts can edit the properties of data objects directly and the dependency graph helps them understand the consequences of this action (such as changes in the views displaying the objects).

To summarize, CzSaw presents four main advantages by

- Capturing and visually representing dependencies among primary and inferred data items.
- Recording an editable sequence of analysis steps, supporting review and editing of an analysis process.
- Visualizing the analysis process through history, script and dependency views, maintaining consistency among these views.
- Enabling higher level analysis through its scripting language.

This paper overviews related work, describes CzSaw and the function of its visual history view, dependency graph and script language, and presents our plans for future work.

2 Related Work

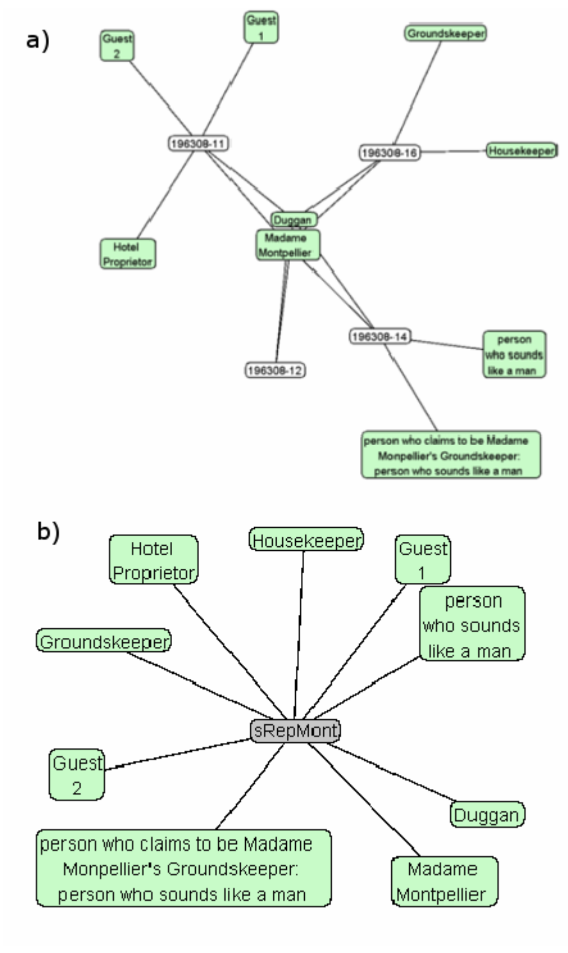
There is an increasing interest in understanding the components of the analysis process, and in capturing and visualizing the sense-making cycles [19, 28].

The first step towards understanding, capturing and visualizing the analysis process is to characterize its elements. In an attempt to do just that, Gotz et al. [11] developed a taxonomy of "semantically meaningful analysis actions". They define these as user interactions with the system which are on a higher semantic level than mouse clicks. Actions such as query, explore and zoom are used to pursue an analytical task like confirming or rejecting a hypothesis. Gotz divides analytical actions into three levels based on intent and defines the middle level as the semantically meaningful level. They categorize these actions and present a scripting language for such interactions, integrated into a visual analytics application, HARVEST, which shows a history trail both in graphical and text representations. Similarly, Amar et al.'s [5] work is focused on user-oriented activities and describes a set of ten low-level analysis task types that capture interactive visual analysis actions. They state that these tasks can be used as a common language in the field and as a measure for the capabilities of the tools. Soo Yi et al. [28] also propose a taxonomy of analytical interaction techniques which are organized around users' intents during their interaction with the system. These research studies lay an initial foundation for CzSaw's scripting language. In CzSaw, elements of the analysis process are defined based on a combination of taxonomies that these studies propose.

Another focus area of CzSaw is providing a visual representation for the history of the analysis process. Such representations appear in many applications such as Adobe Photoshop [2] and Adobe Illustrator [1] which include history capturing mechanisms. These applications record defined user interactions with the system in a linear stream which can be replayed to show the history of development of a piece of artwork. These applications contain a limited memory stack which captures a limited number of interactions and removes older interactions from the bottom of the stack as new interactions are added to the top of it. They also abandon branching history by overwriting the history thread if the user steps back and resumes the interaction from a previous state. Many visual analytics applications also represent the history of user interactions in a visual format. Heer et al. [14] present a method of capturing and showing the history of interactions for Tableau [12] that is similar to CzSaw's visual history view. Each state in their system consists of a VizQL statement and a record of the action taken to reach it. While branches in the history are recorded, unlike CzSaw, only the current branch is shown in the history interface. This interface shows thumbnails of the Tableau visuals which can be bookmarked and annotated. Kreuseler et al. [17] describe a history visualization mechanism for visual data mining, somewhat similar to CzSaw's history view. A tree shows the action hierarchy, with nodes indicating changed views. Like CzSaw, selecting a node activates the node's data view. More relevant to our work is Derthick et al.'s [8] proposed design for representing branches in the history through a tree-view diagram [8]. They designed a branching history mechanism which is applicable to a variety of different software applications – e.g. text editor applications - and includes a tree-structured visualization for navigating across time and scenarios. The visualization also allows browsing the history and selectively undoing/redoing events within a scenario or across scenarios. Robinson et al. [20] propose Re-Visualization (ReVise), a method that captures low-level state changes of the data views. ReVise captures low-level mouse clicks and mouse traces and the history of interaction with the visual analytics tool and shows them as a layer on top of the data view. Unlike ReVise, in CzSaw we capture the history in terms of analytically meaningful statements like "showRelations" and "relatedNodes" instead of recording mouse movements and every single click.

Besides showing temporal history of the analysis process, CzSaw and a number of other research projects visualize the exploratory model of the analysis process and the relationships between the elements and parameters that exist in the dataflow or get generated as a result of exploring the existing elements. VisTrails [23], like CzSaw, is a tool for exploratory visualization, which visually tracks dataflow and workflow changes. It is more of a descriptive representation of a dataflow, whereas the dependency graph in CzSaw is more of a visual representation of how entity collections depend on one another. Bier et al. in their Entity Workspace system likewise support intelligence analysis and sense-making generally through their Think Loop model [6, 19, 25]. CzSaw, like Entity Workspace, helps analysts build an explicit model of important entities (people, places, organizations, phone numbers, etc.) and their relationships. Our focus is to add more control and understanding of the analysis process by adding analysis process views which are built by the process rather than the data. Design systems such as GenerativeComponents [4] support a transaction-based scripting language as the primary system representation. The system records all actions in a human-readable and editable form. Users can (and do) replay and edit the transaction record as an integral part of design work.

Weaver's DEVise [27] uses a relational data model to coordinate multiple views of large datasets. Views and data are linked as logic chains. Users can manipulate multiple coordinated scatter plots interactively using a small number of coordination mechanisms. Jankun-Kelly et al. [16] propose a model to describe the visualization exploration process and a representation to show this model. Their model shows the relations between the elements of the analysis process and how parameters in every sense-making cycle affect the rest of the parameters in the process. Sanfilippo et al. [21], based on data gathered on actual workflows of real analysts, built a system to help uncover scenario content by constructing a "scenario content ontology". In effect their system helps the analyst analyze a document collection, and construct hypotheses from such an analysis, whereas the focus in CzSaw is on visualizations to help analyze the analysis process itself.



a)
Guest 2
Guest 1
Groundskeeper
196308-11
196308-16
Housekeeper
Duggan
Madame Montpellier
Hotel Proprietor
196308-14
person who sounds like a man
196308-12
person who claims to be Madame Monpellier's Groundskeeper: person who sounds like a man
b)
Hotel Proprietor
Housekeeper
Guest 1
person who sounds like a man
Groundskeeper
sRepMont
Guest 2
Duggan
person who claims to be Madame Monpellier's Groundskeeper: person who sounds like a man
Madame Montpellier

Figure 1– Grouping in graph view. a) A graph showing Madame Montpellier and all the documents she is in and the people mentioned in those documents. b) Upon grouping the documents into a single node, we have a much simpler view showing all the people in these documents with the grouped document node at center.

Providing analysts with manual annotating facilities is another aspect of CzSaw. Much research has been done on annotation facilities and how they should be embedded in visual analytics systems to be more convenient for analysts. Jigsaw uses Microsoft OneNote for this purpose. Other approaches include Geotime [10], which allows the analyst to manually add notes to the visualization of the story flow, and Analyst's Notebook [15] which also provides manual capturing utilities for adding charts and notes about the analytical reasoning and sense-making process.

CzSaw builds on the ideas represented in these tools in addition to capturing semantically meaningful interactions with the system in terms of a scripting language. It attempts to build a model of the process itself, via history and dependency-graph views. CzSaw enables analysts not only to view the analysis history; it supports modification and replay of captured analysis sequences. This enables easy comparison of alternate approaches, as well as the re-use of appropriate sequences with different data.

3 Design and Context

CzSaw, like Jigsaw, is aimed at supporting sense-making through investigative analysis of document collections and the entities contained in them.

CzSaw is implemented in Java using a model-view-controller methodology. The data views described in this section were developed by our group, modeled on the data views of Jigsaw. The underlying data representation is somewhat different, however, and is described in the following sections. The current version of CzSaw can read XML files that contain both documents and the entities extracted from them. CzSaw keeps track of every occurrence of an entity in the dataset separately and assigns unique identifiers to each. This allows merging entities if an analyst later believes that multiple occurrences refer to the same entity, as well as splitting entities if they turn out to be different things that appeared with the same name. Like Jigsaw, we consider two entities connected if they appear together in one or more documents. Likewise, two documents are connected if they share at least one entity.

Extracted entities should have a defined type to which CzSaw automatically assigns a colour to help identify the entity type throughout the views. The colour-coding legend is always available for the analyst as a reference.

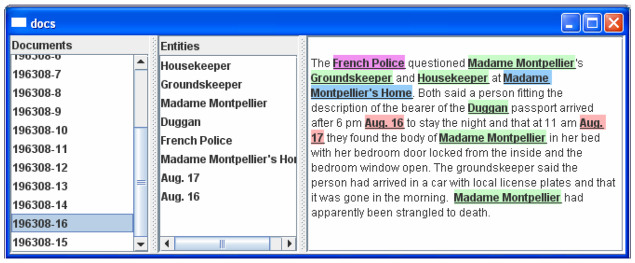
A likely first step in CzSaw is to perform a search. In the search window, the analyst can choose which views display the search results. CzSaw searches among all the extracted entities and then updates the desired views with the results. Inspired by Jigsaw, we provide several data-oriented views, including enhanced graph, lists, and document views.

The graph view is a node-link diagram showing connections between entities and documents. We use the Java Universal Network/Graph (JUNG) library [18] to implement the graph view with a force directed layout. After a layout of the graph is done, the graph is scaled if needed in order to include all graph nodes within the window. Additionally the analyst may display a set of entities or reports as a grouped node instead of several nodes.

Figure 1 demonstrates the grouping technique which is controlled by the analyst and reduces clutter on the screen by hiding details unnecessary to the current analysis.

The list view, also based on one of Jigsaw's views, contains lists of entities and documents with edges used to show connections between items in different lists. A list shows all the entities of one type or can be filtered by a search term. CzSaw adds the capability to move and resize lists, giving analysts the power to view connections between multiple pairs of lists simultaneously. Long scrollable lists often have connections to off-screen list entities and these are shown by an edge connected to the bottom of the visible portion of the list. Buttons at the top and bottom of the list then provide automatic smooth scrolling down or up to the next connected off-screen entity.

The CzSaw document view, like Jigsaw's, displays the full text



The French Police questioned Madame Montpellier's Groundskeeper and Housekeeper at Madame Montpellier's Home. Both said a person fitting the description of the bearer of the Duggan passport arrived after 6 pm Aug. 16 to stay the night and that at 11 am Aug. 17 they found the body of Madame Montpellier in her bed with her bedroom door locked from the inside and the bedroom window open. The groundskeeper said the person had arrived in a car with local license plates and that it was gone in the morning. Madame Montpellier had apparently been strangled to death.

Figure 2 – Document View

of documents (Figure 2). The left panel shows the set of documents currently being viewed while the right shows the text of the current document with entities highlighted. The middle panel shows a list of entities in the document currently being viewed. The document view is useful because reading the documents provides the context of a connection between entities.

Communication among views is essential in CzSaw. Each view updates based on changes made in other views, providing a foundation for brushing and linking, and higher level visual analytics techniques. Internally, CzSaw treats views as links between interface objects and underlying data and thus view creation is explicitly recorded in the script. Thus an analyst can review and change how views relate to data, with the implication that views persist across sessions.

3.1 Analysis Process Views

Prompted by the oft-repeated request for overviews of the analysis process ("I need to see the state of the analysis yesterday noon when I was last working on…"), CzSaw has new process and model-based views that both visualize the ongoing analysis model, and allow direct interaction with that model. These views are also motivated by the success of systems like Gotz' HARVEST which attempts to capture the semantics of user interactions, and in particular by the success of systems in the computer-aided design area [3] to show a 'generative' view of the development of an underlying model of the analysis process.

This set of views of the analysis model is based on an underlying 'semantic' level representation of the sequence of user actions. It comprises three related views and the objects they represent:

- *Script view:* a view of the captured script statements in a reviewable, re-playable and editable manner.
- *History view:* a visual history of the analysis process; a collection of the temporal sequence of data views. The history view is also the place where an analyst can add notes, in a manner similar to Heer et al. [14].
- *Dependency graph:* a special graph view showing as nodes the various entities, collections of entities etc. generated during the analysis, with links indicating dependencies.

We describe each of the analysis model views in the following subsections.

3.2 Scripting driven propagation system

The process of investigative analysis can be compared to the process of writing a story. In this case the story is the analysis process and its components are the steps taken in CzSaw, so reading the story may help an analyst better understand how and why they arrived at a result and how accurate or significant it might be.

Based on this idea, we defined a scripting language for performing operations meaningful to the analysis process. The language consists of commands that a user may enter directly. Users of course also interact directly with data views, and, analogous to HARVEST [11] each such user interaction is translated into a block of one or more commands that we call a transaction. Thus analysts perform the analysis process by either interacting with views or by directly typing script commands.

Transactions are at the level of meaningful tasks completed in the system; script commands are programming elements that implement transactions. Only actions which modify the visual or data model of the system are recorded. Thus no script commands are generated based solely on non-dragging mouse movement or the typing of a word in the search window.

When the analyst presses the search button then a block of commands is recorded. Some script commands such as "search" and "relatedNodes" describe system actions on the data structures. Other commands describe system actions that directly result in data view changes such as "showNodes" and "hide". Commands in this second group will be interpreted differently depending on the particular data view. An example is given in section 3.2.1.

The script is not just a passive recording of the system actions. The script is actually what drives CzSaw. Rerunning this script step by step will replay the analysis process. The data views only update based on the script regardless of whether the commands were generated from an analyst's interactions with the data views or typed directly in. One way of thinking of this script is as an application created by the analyst while using CzSaw. Editing it allows quick refinement of the analysis process, since any later steps after the edit do not have to be retaken by the analyst. In this way, the script can be modified to see alternatives or fix mistakes. An important aspect of the script is that it makes analysis interactions reusable, especially since analysis processes often consist of many repetitive actions [22]. Additionally an analyst may find exploration patterns applicable to different datasets or an updated version of the same dataset. Such tasks are easily accomplished by simply extracting the appropriate set of script commands and running them on different datasets. Suitably tested script blocks should decrease errors in sequences of analytical actions.

Within a script, new data can be added, such as adding a new report just received by the analyst. In the real world, data is dynamic, growing and changing. Current analysis results may change, or road blocks be removed. Adding new data and rerunning the script allows adaptation to such data changes which can reveal new results. Also it is possible to explore alternatives by adjusting parameters in the script. For example, an analyst could iterate a predefined analytical process over a list of people.

In CzSaw, experienced analysts who wish to directly program part of the script may do so within the script view. The script view is where the analyst can create, copy, edit and run script commands. Thus CzSaw enables a much greater control over the analysis process itself than would be possible using only the data views.

3.2.1 Example Scenario

We present an example analytical scenario derived from the novel *The Day of the Jackal,* as follows. In the story, an assassin has been secretly hired in 1963 to kill President De Gaulle. Inspector Lebel is trying to discover the assassin's identity. Through British police, Lebel learns that a passport has been issued to Paul Duggan, who died 30 years earlier. A week after this discovery, a person bearing the Duggan passport enters France. The next day, Duggan is discovered to have stayed at a hotel in southern France. Police interview the hotelier, and discover that "Duggan" left with Madame Montpellier. Madame Montpellier, when interviewed admits only an affair with Duggan. The next day, Madame Montpellier is discovered murdered. Lebel now has a suspected murderer (Duggan), and his description, but no name, since the assassin assumes a new identity.

The analyst now needs to discover the new identity of the assassin. A pattern that the assassin uses is to stay with people he befriends so that he does not have to give identity documents to hotels. Another analytical pattern is the "fake passport" pattern, in which a passport is obtained for someone who died as a child. In the real world scenario, we can not simply look for "fake passport" because there may be many people bearing a fake passport that are irrelevant to this case. Therefore, we start from the most significant event in the data - Madame Montpellier's murder and see what connections we can find.

CzSaw's search function is available both in the graphical user interface (GUI) and in the scripting language. The script form of a search for person "Montpellier" is:

entPerMont = search(Montpellier, person)

The left side of this equation is the variable name which can be used by later commands. It contains a list of entities that have type "person" and value "Montpellier". Below is the full script of the investigation from Madame Montpellier to the fake passport bearer organized by transaction, interlaced by screenshots of the graph view (Figure 3, Figure 4, Figure 5, Figure 6, and Figure 7) which were captured while executing this script. Also included are descriptions of how the same actions could be taken by direct user interaction with the graph view.

Transaction 1:
gView = newView(GraphView)
Transaction 2 (Figure 3):
entPerMont = search(Montpellier, person)
sPerMont = showNode(entPerMont, gView, GRAPH)

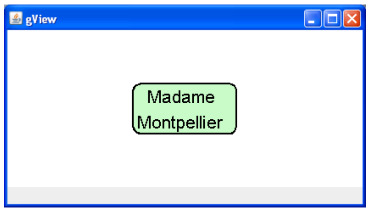


Figure 3 – In a new graph view, show search results for people named "Montpellier".

Transaction 3 (Figure 4):
repMont = relatedNodes(entPerMont, report)
relPLMont = relations(entPerMont, repMont)
sRepMont = showNode(repMont, gView, GRAPH)
sRelPLMont =showRelations(relPLMont, sPerMont, sRepMont)

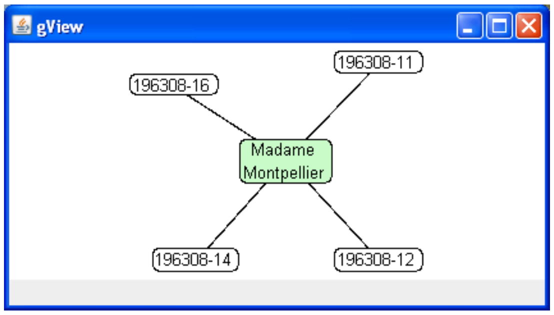


Figure 4 - Get all related reports for "Madame Montpellier" by expanding this node in graph view.

Transaction 4 (Figure 5):
entPerRelMont = relatedNodes(repMont, person)
relPerMont = relations(entPerRelMont, repMont)
sEntPerRelMont = showNode(entPerRelMont, gView, GRAPH)
sRelPerMont =showRelations(relPerMont, sEntPerRelMont, sRepMont)

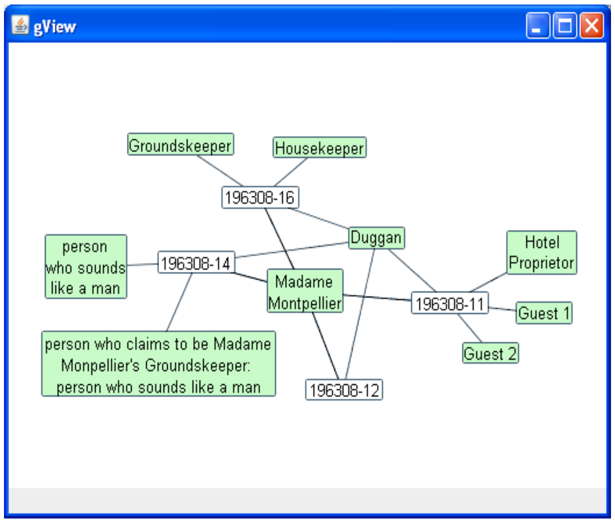


Figure 5 - Find all persons within these reports. In the graph view this is done by choosing to expand each report node only for type person (through right-click menu). It looks like Duggan is the person to be examined further.

Transaction 5:
entPerDuggan = search(Duggan, person)
sPerDuggan = showNode(entPerDuggan, gView, GRAPH)
Transaction 6 (Figure 6):
repDuggan = relatedNodes(entPerDuggan, report)
sRepDuggan = showNode(repDuggan, gView, GRAPH)
relDuggan = relations(entPerDuggan, repDuggan)
sRelDuggan = showRelations(relDuggan, sPerDuggan, sRepDuggan)

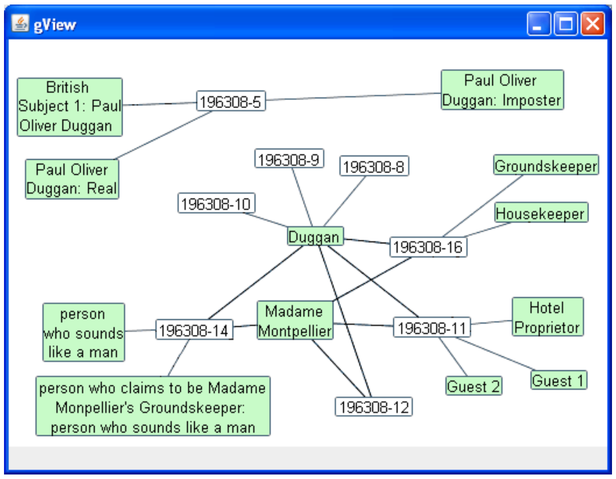

gView
British Subject 1: Paul Oliver Duggan
196308-5
Paul Oliver Duggan: Imposter
Paul Oliver Duggan: Real
196308-9
196308-8
Groundskeeper
196308-10
Housekeeper
Duggan
196308-16
person who sounds like a man
196308-14
Madame Montpellier
196308-11
Hotel Proprietor
person who claims to be Madame Monpellier's Groundskeeper: person who sounds like a man
Guest 2
Guest 1
196308-12

Figure 6 – Search for Duggan and find all the related reports. All related reports could be found by expanding each Duggan node.

Transaction 7 (Figure 7):
entPerRelDuggan = relatedNodes(repDuggan, person)
sEntPerRelDuggan = showNode(entPerRelDuggan, gView, GRAPH)
relPerDuggan = relations(entPerRelDuggan, repDuggan)
sRelPerDuggan = showRelations(relPerDuggan, sEntPerRelDuggan, sRepDuggan)

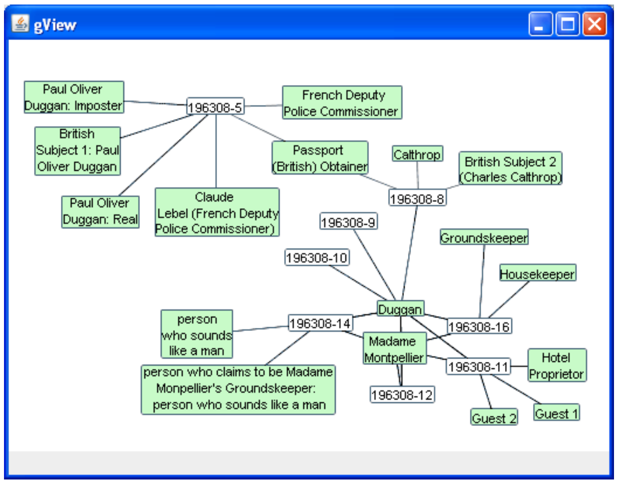

gView
Paul Oliver Duggan: Imposter
196308-5
French Deputy Police Commissioner
British Subject 1: Paul Oliver Duggan
Passport (British) Obtainer
Calthrop
British Subject 2 (Charles Calthrop)
Paul Oliver Duggan: Real
Claude Lebel (French Deputy Police Commissioner)
196308-8
196308-9
Groundskeeper
196308-10
Housekeeper
Duggan
person who sounds like a man
196308-14
196308-16
Madame Montpellier
Hotel Proprietor
person who claims to be Madame Monpellier's Groundskeeper: person who sounds like a man
196308-11
196308-12
Guest 2
Guest 1

Figure 7 - All related reports and people: people in reports with Duggan can be found in the graph view in the same manner as in Figure 5. “Charles Calthrop” may be the suspect.

The above script can be saved to a text file. CzSaw can import this file to be replayed on a new set of data. In the real world, new reports will continue to arrive as the investigation progresses. Rerunning this script will reflect the most up-to-date information to show the new reports and new entities. Figure 8 shows this in a retrospective way – suppose we were in the middle of the investigation, and some report were not there yet (report 196308-8 does not yet exist).

In CzSaw, analysts can also modify and edit the script, e.g. by varying parameters in the command to explore different results. After the changes in the dataset or the script, the whole script may be unable to run completely. For example, some reports may have been removed if they were found to be untrue. Then their related entities turn into empty (null) results, and this may also prevent subsequent operations from continuing. We argue that this is not necessarily a bad thing for the analysis process. It actually warns the analyst that either their hypothesis is wrong, or there is some information missing that needs to be investigated further.

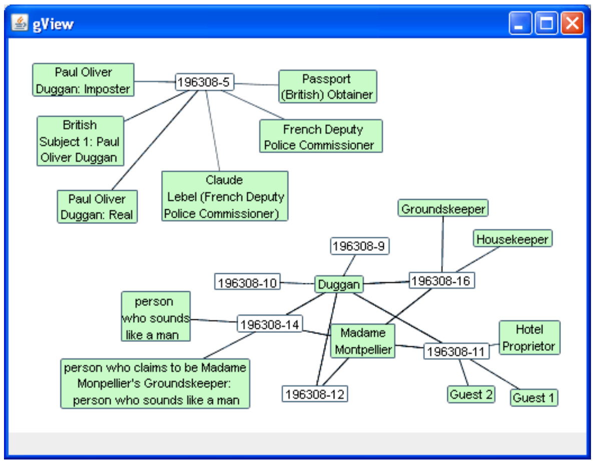

gView
Paul Oliver Duggan: Imposter
196308-5
Passport (British) Obtainer
British Subject 1: Paul Oliver Duggan
French Deputy Police Commissioner
Claude Lebel (French Deputy Police Commissioner)
Paul Oliver Duggan: Real
Groundskeeper
Housekeeper
196308-9
196308-10
Duggan
196308-16
person who sounds like a man
196308-14
Madame Montpellier
Hotel Proprietor
196308-11
person who claims to be Madame Monpellier's Groundskeeper: person who sounds like a man
196308-12
Guest 2
Guest 1

Figure 8 - Result of running the script with a report missing.

Analysts do not need to know anything about scripting as all the above work can easily be accomplished through direct interactions with the data views. The script will be generated based on these interactions with CzSaw. However, we believe, for the expert user, this scripting ability will give them a more direct, precise, and easier way to work on complex tasks.

3.3 Visual History

Sense-making is an iterative process and analysts add to their understanding about data through multiple cycles of analytical reasoning [22]. During the analysis process, analysts may realize that a particular line of reasoning needs to be changed, and they need to step back and adjust the process. The visual history view supports adjusting the analysis process by allowing it to be resumed from any state in the current analysis history.

The visual history view shows the history of the analysis process in temporal order. This view provides an overview of the analysis paths to help analysts understand where they are in the analysis process. Zooming into a history node shows details of the visible data views at that time. Whenever analysts commit model-altering interactions with the system – interactions that have an equivalent transaction (sequence of commands in our scripting language) – we capture a screenshot of the state of all of the open CzSaw data views. These screenshots are time-stamped and appear in the history view window in a linear layout. Each node in this history view corresponds to the state of the system at the end of completing a script transaction. Whenever the analyst needs to step back to adjust the analysis process, or when he or she needs to step back to explore an alternative analytical avenue, they first find the history node at which they wish to try a different analysis path. Then, selecting that history node triggers CzSaw’s script engine to replay the generated script of the analysis process from the beginning to the end of the selected node’s transaction. This will update the state of all of the views to the state that they had at the selected time. This method (running the script from the beginning every time that a new branch is added) is relatively slow and we are working on improvements.

Resuming the analysis process from a node causes a new line of history nodes to start from that node as a branch. Each branch shows an alternative path in the history. Figure 9 shows a sample history view window. The top row shows the first branch of the history. Each branch line is displayed from left to right. The first node on the left of the top row shows the state of the system at the time which is stamped on the node. From the small dependency graph on this first thumbnail we can see that the analysis process has just started. The second node of the top row shows the analyst has generated more script by opening an empty graph window (which has been added as a node in the dependency graph). The third node on the top row shows that the analyst has shown a person in the graph view (by searching for a person). The top row ends here because the analyst stopped the analysis process or started a new branch. In this case, we can see that the analyst has clicked on the second node of the top row and started a new branch. The first node on the second row shows a different graph diagram, one which contains more graph nodes (as this new search returned more results). Arrows highlight the direction that nodes are added to the view (Left to right, top to bottom) to add to the readability of the diagram. Analysts can annotate any history node to keep track of their decision-making. They can also bookmark nodes in order to highlight decision points. An icon appears beside each node that has an annotation or bookmark.

Analytic reasoning includes numerous interactions with CzSaw. This will cause the history view to grow very rapidly, which makes the visualization harder to follow. Addressing this issue while minimizing the use of screen space is a challenging task. Although initially we use a variant of a simple pan and zoom, we plan to supplant this with a variant of the continuous zoom algorithm [9], a hierarchical detail-in-context method. We hope that showing detail and context of the visual history may help analysts more easily recall “where they were” in longer analysis processes on resumption after a hiatus or distraction.

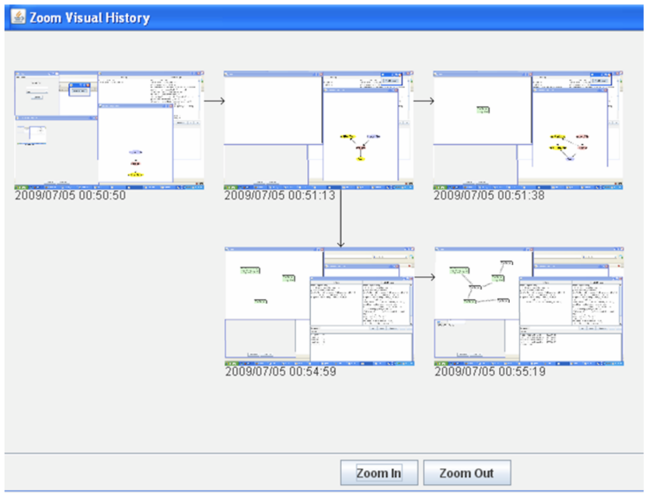


Figure 9 – CzSaw adds nodes to the History View from left to right in a linear layout. New branches start on new lines under the node from which they are branched. The history nodes are time-stamped and the arrows show the direction of the analysis process.

3.4 Dependency Graph

As described before, an important goal of CzSaw is to capture and visualize a model of the analysis process, so the analyst can interpret and edit it—we view this explanation of an analytical process and result as a major product of analysis. As part of supporting this goal, we developed a "dependency graph" view, a node-link diagram of the relationships among the results of every analytical interaction. The design of this view was inspired by the Symbolic Model view in Bentley's parametric CAD application GenerativeComponents [3] in which a node-link diagram shows components as nodes while edges show dependencies between them. The symbolic diagram not only helps users understand the logic structure of the 3D model, but more importantly it serves as an additional and new view of the design of the analysis story.

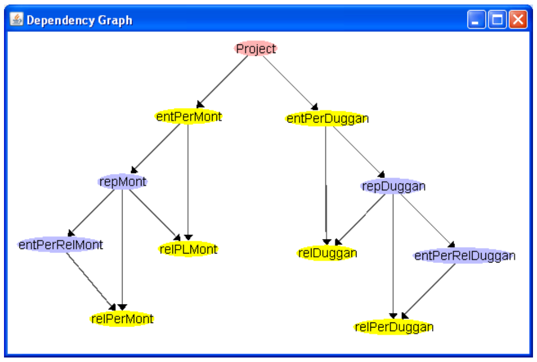


Figure 10 - Dependency Graph – based on the script in section 3.2.1

Figure 10 shows the dependency graph for the script in section 3.2.1, where variables are shown as nodes and dependency relationships as directed edges. This graph maintains the integrity of objects during the analysis process: when a variable's (node's) value is changed, descendent values will be updated automatically. The root node "project" represents the original data and project name; "entPerMont" and "entPerDuggan" come from the original data and were created by the search action; "repMont" is based on "entPerMont; "relPLMont" is derived from "repMont" and "entPerMont", etc. The dependency graph thus helps users better understand the logic underlying the steps taken during the analysis process and what consequences may occur from editing this process.

We believe the combination of dependency graph, visual history and script together will help the analyst gain insight into the analysis process itself. The dependency graph represents the current state of the analysis process as an active graph of objects and relations. It represents an "internal" view of the analysis process whereas the history view represents an external view.

4 Discussion and Future Directions

We believe that the current CzSaw prototype is a good start toward our goals of capturing and visualizing the analysis model, of providing a 'design pattern' capture and reuse capability for advanced analyst-developers, and of course providing effective, interactive data visualizations for investigative analysis. We will determine how well we achieve these goals through user studies, though integration of key features will be needed first. Our main contribution is in the development of a system that provides support for capturing the semantic structure of an analysis session, and in making this visible to and interactively modifiable by the analyst. While we have established the basic framework, clearly much development is needed, and many research questions remain to be explored.

Our scripting language for example, though potentially powerful, is still at a primitive stage and needs much development. More importantly we need to design and carry out the experiments necessary to validate and tune our choices of language elements and structure. After this we intend for the script to become a fully functioning programming language with conditional control, iteration, etc. Use of these advanced functions may replace repetitive analytical actions taken as part of the sense-making process. Currently repetitive action can be repeated by replicating script transactions and changing the variables used.

There are several scalability issues to address as well. For example the graph structure of a visual history grows rapidly and we will need to use hierarchical detail-in-context methods to allow the analyst to navigate the analysis history without losing their place. We also need to consider issues of dataset size: CzSaw works with tens to hundreds of documents with a limited number of entities but is less effective with larger numbers and we need ways to alleviate these issues.

Note taking is currently limited to notes added to nodes in the history view. It might be distracting to move between data and history views to add notes and some analysts may still find using paper and pen more convenient. We are investigating other ways for convenient note taking.

One area of future research concerns combining data views to form a hybrid view that gives the analyst a greater ability to control the level of semantic zoom of each part of the visualization. For example bringing lists into a graph view should have advantages over both graph view and list view. The combination of views to form one generic view also simplifies the rules of the script language, as the single view will be able to handle any visual item.

5 Conclusion

Various analyses of the analytic reasoning and sense making process with visual analytics tools suggests there are a limited number of semantically meaningful actions that users of these systems take; see for e.g. [16, 17]. Looking at patterns of these actions may provide helpful information to analysts about the analysis process. To this end, we have presented CzSaw, a visual analytics tool focused on capturing the analysis process and visualizing it. CzSaw includes several data views that allow analysts to explore document collections and better understand connections between entities in the documents. Within this system we have developed a scripting language covering many semantically meaningful analytic interactions and illustrating them both in temporal and dependency sequences. Analysts may extract and run script transactions on different datasets to perform common tasks quickly or adapt to new data. The idea behind CzSaw is to give the clinician greater support in navigating their past analyses and controlling their future processes.

ACKNOWLEDGEMENT

This work is supported in part by grants from The Boeing Company ("Visual Analytics in Canada"), NSERC (STPG 336506, "Visual Analytics for Safety and Security"), NSERC Discovery Grants and NSERC PGS, and we gratefully acknowledge this support.

We also thank Professor John Stasko and his group at Georgia Tech for kindly providing us with a running version of Jigsaw.

REFERENCES

[1] Adobe Systems Incorporated. Illustrator. www.adobe.com/products/illustrator/?promoid. Accessed 15 July 2009.

[2] Adobe Systems Incorporated. Photoshop. www.photoshop.com. Accessed 15 July 2009.

[3] R. Aish.. "Bentley's GenerativeComponents: a Design Tool for Exploratory Architecture", *Bentley Systems Inc.* 2003.

[4] R. Aish and R. Woodbury, "Multi-level Interaction in Parametric Design," *Lecture Notes in Computer Science*, vol. 3638, p. 151, 2005.

[5] R. Amar, J. Eagan, and J. Stasko, "Low-Level Components of Analytic Activity in Information Visualization," *Proceedings of the IEEE Symposium on Information Visualization*, pp. 111–117, 2005.

[6] E. Bier, S. Card and J. Bodnar. "Entity-Based Collaboration Tools for Intelligence Analysis". In *IEEE VAST 2008*, pp. 99-106, 2008.

[7] A.M. Cuadros, F.V. Paulovich, R. Minghim, and G.P. Telles, "Point Placement by Phylogenetic Trees and Its Application to Visual Analysis of Document Collections," *IEEE Symposium on Visual Analytics Science and Technology 2007 (VAST 2007)*, pp. 99–106, 2007.

[8] M. Derthick and S.F. Roth, "Data Exploration across Temporal Contexts," *Proceedings of the 5th international conference on Intelligent user interfaces*, ACM New York, NY, USA, pp. 60-67, 2000.

[9] J. Dill, L. Bartram, A. Ho, and F. Henigman, "A Continuously Variable Zoom for Navigating Large Hierarchical Networks," *Systems, Man, and Cybernetics, 1994. 'Humans, Information and Technology', 1994 IEEE International Conference on*, vol.1 pp. 386-390, 1994.

[10] R. Eccles, T. Kapler, R. Harper, and W. Wright, "Stories in Geotime," *Information Visualization*, vol. 7, pp. 3-17, 2008.

[11] D. Gotz and M. Zhou. "Characterizing Users' Visual Analytic Activity for Insight Provenance," *Proc. IEEE Symposium on Visual Analytics Science and Technology*, pages 123–130, 2008.

[12] P. Hanrahan, "Tableau Software White Paper-Visual Thinking for Business Intelligence," *Tableau Software, Seattle, WA*, 2003.

[13] Hao, M.C., Dayal, U., Keim, D.A., Morent, D., and J. Schneidewind, "Intelligent Visual Analytics Queries", *Processings of the IEEE Symposium on VAST2007*, Sacramento, CA, USA, pp. 91-98, 2007.

[14] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala, "Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1189-1196, 2008.

[15] i2 Incorporated. Analyst's Notebook. www.i2inc.com. Accessed 15 July 2009.

[16] T.J. Jankun-Kelly, K.L. Ma, and M. Gertz, "A Model and Framework for Visualization Exploration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 357-369, 2007.

[17] M. Kreuseler, T. Nocke, and H. Schumann, "A History Mechanism for Visual Data Mining," *IEEE Symposium on Information Visualization, 2004. INFOVIS 2004*, pp. 49-56, 2004.

[18] J. O'Madadhain, D. Fisher, and T. Nelson. JUNG: Java Universal Network/Graph Framework. http://jung.sourceforge.net/. Accessed 15 July 2009.

[19] P. Pirolli and S. Card, "The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis," *Proceedings of International Conference on Intelligence Analysis*, 2005.

[20] A.C. Robinson, C. Weaver, and G. Center, "Re-Visualization: Interactive Visualization of the Process of Visual Analysis," *Workshop on Visualization, Analytics & Spatial Decision Support at the GIScience conference*, 2006.

[21] A. Sanfilippo, B. Baddeley, A. J. Cowell, M. L. Gregory, R. Hohimer and S. Tratz. "Building a Human Information Discourse Interface to Uncover Scenario Content", In *the 2005 International Conference on Intelligence Analysis,* https://analysis.mitre.org/proceedings_agenda.htm, accessed on 29 March 2009.

[22] Y. Shrinivasan and J.J. van Wijk, "Supporting Exploration Awareness for Visual Analytics", *Processings of the IEEE Symposium on VAST2008*, Columbus, Ohio, USA, pp. 185-186, 2008.

[23] C.T. Silva, J. Freire, and S.P. Callahan, "Provenance for Visualizations: Reproducibility and Beyond," *Computing in Science & Engineering*, vol. 9, pp. 82-89, 2007.

[24] J. Stasko, Görg C, Liu Z, Singhal K. "Jigsaw: Supporting Investigative Analysis through Interactive Visualization", *Proceedings of IEEE Symposium on Visual Analytics, Science, and Technology (VAST'07)* (Sacramento, CA), IEEE Computer Society Press: Silver Spring, MD; 131–138, 2007.

[25] J.J. Thomas and K.A. Cook. "Illuminating the Path", IEEE Computer Society: Chicago, 2005.

[26] C. Ware, A. Gilman, and R. Bobrow, "Visual Thinking with an Interactive Social Network Diagram.", *Diagrams 2008*, Herrsching, Germany, pp 118-126, 2008.

[27] Chris Weaver, "Metavisual Exploration and Analysis of DEVise Coordination in Improvise," *Fourth International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV'06),* pp.79-90, 2006.

[28] Ji Soo Yi, Youn ah Kang, John T. Stasko and Julie A. Jacko, "Toward a Deeper Understanding of the Role of Interaction in Information Visualization", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 13, No. 6, pp. 1224-1231, November/December 2007.