Construction and Usage of the Semantic Interaction Pipeline

Unpublished
XX(X):1–29

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Michelle Dowling, John Wenskovitch, Peter Hauck, Adam Binford, Theo Long, Nicholas Polys, Chris North

Abstract

Semantic interaction techniques in visual data analytics allow users to indirectly adjust model parameters by directly manipulating the visual output of the models. Many existing tools that support semantic interaction do so with a number of similar features, including using an underlying bidirectional pipeline, using a series of statistical models, and performing inverse computations to transform user interactions into model updates. We propose a visual analytics pipeline that captures these necessary features of semantic interactions. Our flexible, multi-model, bidirectional pipeline has modular functionality to enable rapid prototyping. This enables quick alterations to the type of data being visualized, models for transforming the data, semantic interaction methods, and visual encodings. To demonstrate how this pipeline can be used, we developed a series of applications that employ semantic interactions. We also discuss how the pipeline can be used or extended for future research on semantic interactions in visual analytics.

Introduction

Semantic interaction* is a powerful user interaction methodology, allowing analysts to explore and discover relationships in data[1]. Semantic interaction exploits intuitive interactions to manipulate underlying model-level parameters. Through semantic interactions such as the direct manipulation of visualizations, the system is able to learn about the analyst's reasoning process and thereby modify underlying models, which in turn alter the visualization with this newly-learned information[2]. This coupling of machine learning algorithms with user knowledge and interactions allows for the creation of robust analytics systems that collaboratively exploit the skills of both human and computer.

For example, a number of semantic interaction tools and techniques have been developed that make use of a visual "proximity ≈ similarity" metaphor to map observations into a visualization[2–7]. As analysts manipulate the observations in these visualizations, they communicate to the system a desired similarity between a subset of observations, which in turn updates the underlying model parameters that define the visualization. These interactions allow a user of the system to continue exploring and understanding relationships in the data without pausing to manipulate model parameters manually. This frees the analyst's cognition to focus on high-level analysis concepts rather than low-level parameter details[8]. As the analyst continues to perform such semantic interactions, the system learns more about the analyst's reasoning, and the visualization incrementally adjusts to reflect the data explorations that the analyst is investigating[3].

Though a number of tools that use semantic interactions have been developed, each is limited to its respective design goals. For example, Andromeda[6] uses a Weighted Multidimensional Scaling (WMDS) algorithm to visualize high-dimensional quantitative data, but includes no support for foraging new data. Conversely, StarSPIRE[7] and its BigSPIRE[10] extension use a multi-model approach to forage for and potentially display thousands of documents, but these tools are limited to text data. These tools each implement a specific pipeline to answer targeted research questions. Although they work well for their designed use cases, the manner in which the systems are implemented makes experimenting with different data types, mathematical models, semantic interactions, and visual encodings difficult. Thus, further research into visual analytics tools that enable semantic interactions with these specific tools is limited.

To enable the exploration of many forms of semantic interaction, we begin by defining the characteristics of a new flexible pipeline for semantic interaction-enabled tools. We then define components of a generalizable semantic interaction pipeline that correspond to these characteristics. We demonstrate the flexibility of the pipeline by prototyping a variety of visual analytics tools. We conclude by discussing how these prototypes, and therefore this semantic interaction pipeline, can be used to explore data types, mathematical models, interactions, and visual encodings in semantic interaction-enabled visual analytics tools.

We note the following contributions:

1. We create a stricter definition of semantic interaction, expanding on the principles defined by Endert et al[3] and using V2PI as motivation.
2. Using V2PI-enabled tools, we identify requirements for a pipeline for bidirectional interaction with visual analytics models, and implement this pipeline to enable rapid prototyping of visual analytics tools.
3. We supply five example applications using this semantic interaction pipeline to explore different data

*For readers unfamiliar with semantic interaction and multidimensional projection, we provide a collection of terms and definitions in Table 1.

Term	Definition
Attribute	A property or dimension of an observation. This can be thought of as a single cell in a data table.
Dimension Weight	A weight on a dimension that is representative of both the level of importance the analysts places on that dimension and the influence that dimension has on the visualization.
Foraging Loop	The combination of analytic steps from the Sensemaking Process that are used to search for new information.
Model	A mathematical algorithm that manipulates a set of data according to model parameters. This definition of a model is the generalization of the Model pipeline component defined in Table 2.
Model Parameters	The parameters that define how data is manipulated by a model.
Observation	A single item in a dataset. This can be thought of as a single row in a data table.
Pipeline	The combination of the raw data, models used to manipulate the data, and the final visualization produced based on the manipulated data. This definition of a pipeline is the generalization of the Pipeline defined in Table 2.
Semantic Interaction	An interaction within a visualization that requires updating one or more of the underlying models that generated the visualization.
Sensemaking Loop	The combination of analytic steps from the Sensemaking Process that are used to synthesize information.
Sensemaking Process	The conceptual flow of data defined by Pirolli and Card[9] that is used to transform information from raw data into a theory through a series of analytics steps (as shown in Figure 2.
Visualization	An interactive, visual representation of data. This definition of a visualization is the generalization of the Visualization pipeline component defined in Table 2.
V2PI	A statistical semi-supervised machine-learning methodology developed by Leman et al.[5] for realizing semantic interaction.
Weighted Multidimensional Scaling (WMDS)	A mathematical algorithm to determine a low-dimensional projection of high-dimensional data based on weighted high-dimensional distances of the data.

Table 1. Semantic interaction and multidimensional projection terms and definitions.

types, mathematical models, semantic interactions, and visual encodings.

The Design Space of Semantic Interaction Pipelines

Wang et al. present a survey of visual analytics pipelines, identifying commonalities and differences between those pipelines[11]. They use the visual analytics model provided by Keim et al.[12] to motivate the initial steps of the survey. As seen in Figure 1, Keim et al. capture the process behind visual analytics tasks at a high level, but they abstract any details of the computational model(s) and visualization(s) into single nodes in the graph. Thus, the precise mechanisms used to process and visualize the data are not adequately captured in this model. This particular visual analytics model is insufficient for depicting how to create visual analytics tools that support semantic interaction.

To address this lack of detail in current visual analytics pipelines, our goal in this work is to capture the structure of the feedback loop between the the different data processing components of the visual analytics pipeline. We accomplish this goal by first defining identifying the requirements of a semantic interaction-enabled visual analytics pipeline, which thereby allows us to design a new generalized pipeline that captures interactions between the pipeline components.

Because our motivation for defining this new pipeline lies in the desire for semantic interactions, in this section we first discuss sensemaking, the driving force behind semantic interaction. We then discuss a particular realization of semantic interaction, Visual to Parametric Interaction

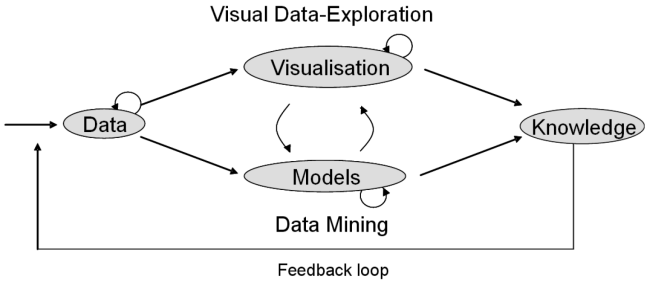

Figure 1. The visual analytics model provided by Keim et al.[12] provides a high-level overview of the structure of visual analytics knowledge discovery, but lacks detail in defining the feedback loop between Models and Visualization. In order to support semantic interaction, additional structure is required in a more detailed pipeline.

(V2PI), which is simple for non-experts of the system to understand and use, yet is interpreted as a complex manipulation of the underlying visualization models. From there, we exemplify the principles of semantic interaction through StarSPIRE, Andromeda, and a collection of other visual analytics tools. We conclude that there are three common properties that enable semantic interaction in visual analytics tools: (1) a series of models, (2) looping foraging and sensemaking interactions, and (3) inverse processes.

Sensemaking

Pirolli and Card[9] introduce a conceptual data flow (Figure 2) showing the transformation of information from raw data into a theory through a series of analytic steps. Each of these

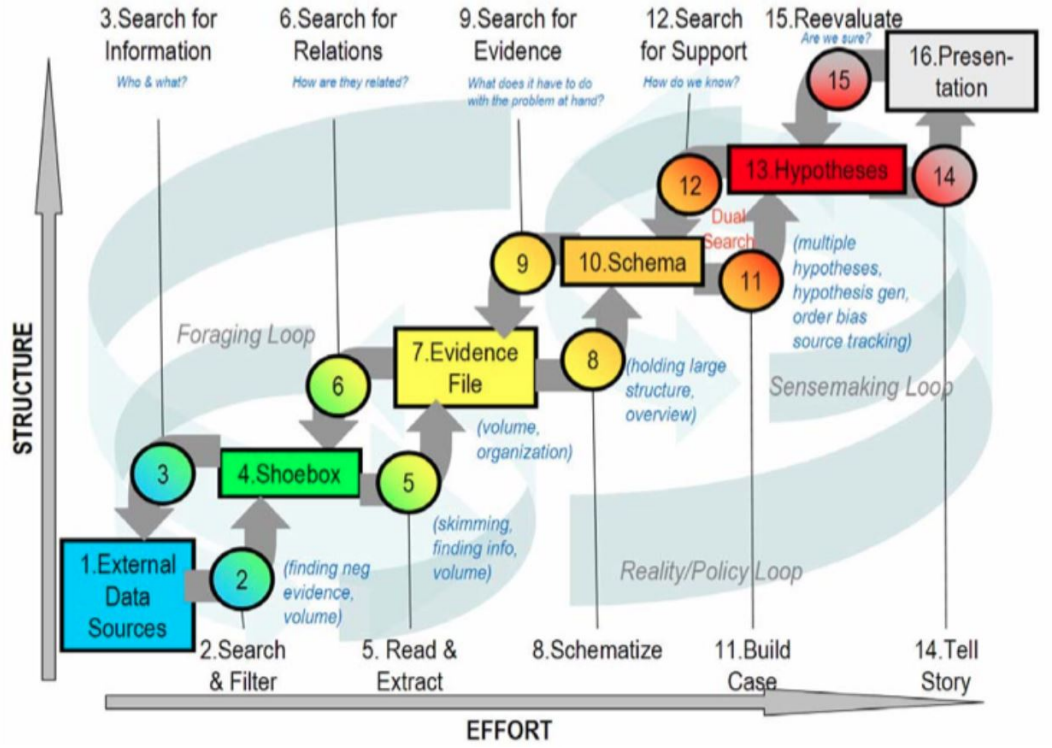

Figure 2. In the Sensemaking Process[9] the desired sensemaking interaction contains a backward (or inverse) process for each forward step. Chaining these combined forward/inverse processes as composable processes yields a full bidirectional pipeline.

steps is in the form of a compact bidirectional loop. For example, an initial transfer of information from the external data to the "shoebox" may uncover a new entity that was not previously recognized. The existence of this new entity then causes the analyst to return to the external data to uncover new information in support of that entity.

These compact loops chain together to form two major bidirectional loops. The first loop is the foraging loop[13], described as involving "processes aimed at seeking information, searching and filtering it, and reading and extracting information." Following the foraging loop is the sensemaking loop[14], described as involving "iterative development of a mental model (a conceptualization) from the schema that best fits the evidence." We will refer to this overall conceptual data flow as the Sensemaking Process.

A number of tools have been implemented to support the Sensemaking Process. One approach is to develop semantic interactions that are specifically designed to support foraging and/or sensemaking tasks. As our focus is on semantic interactions, we next introduce V2PI, followed by several tools that implement V2PI.

Visual to Parametric Interaction (V2PI)

V2PI[5] provides a statistical semi-supervised machine-learning methodology for realizing semantic interaction. It is a framework for creating interactive visualizations that rely on both proven statistical methods and expert user judgment. Analysts are afforded the ability to both learn from visualizations and adjust them directly to inject feedback.

The bidirectional framework formulated in V2PI is shown in Figure 3. In this framework, a visualization V is created by processing data D and parameters θ through a mathematical model M. This visualization is presented to the user U (the

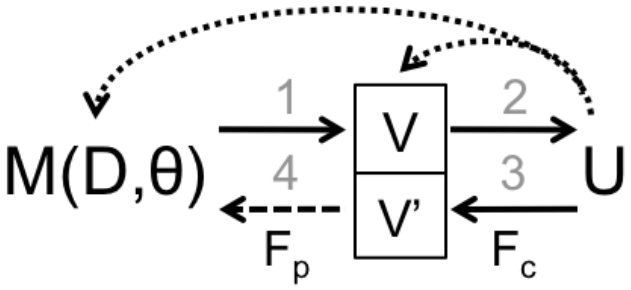

Figure 3. V2PI[5] is a mathematical formulation of bidirectional functionality. This framework supports the creation and manipulation of a visualization V, which in turn enables the manipulation of the parameters θ that influence model M. The parameterized feedback (F_p) represents an inverse process similar to what is included in the Sensemaking Process, in which the user interaction is interpreted as a set of updates to model parameters.

analyst) to evaluate. The analyst can directly manipulate the visualization, referred to as cognitive feedback (F_c), which creates an updated visualization V'. This cognitive feedback is formalized into parameterized feedback (F_p), which updates model M through new parameters θ.

Leman et al. demonstrate the following example of V2PI in their paper[5]. A cities dataset with 25 observations, 10 real dimensions (including latitude and longitude, among others), and 20 noise dimensions is projected using WMDS. In the initial projection with uniform weights across the dimensions, the cities are scattered about the projection. An analyst then provides cognitive feedback (F_c) to the projection by selecting five of the cities and dragging them to approximately their correct relative geographic

locations. The system interprets this updated visualization (V') as parameterized feedback (F_p), solving for new WMDS dimension weights (θ) that increase the weight on the latitude and longitude dimensions while decreasing the weight on all others. A new visualization (V) is formed from the updated weights, roughly positioning all of the cities in their relative geographic locations. In this manner, both the noise dimensions and the "real" dimensions that were unnecessary to the analyst's current line of inquiry had minimal impact on the new projection. Instead, the important latitude and longitude dimensions were uncovered by parameterizing the analyst's cognitive feedback via V2PI.

The fact that the analyst's interaction with the visualization itself is translated by the system into updated model parameters qualifies V2PI as a type of semantic interaction. From the Sensemaking Process perspective, this translation on behalf of the analyst also means that the analyst can remain within their sensemaking loop rather than deviating to manipulate the model parameters themselves. The following subsections describe tools and techniques that use V2PI and demonstrate the principles of semantic interaction.

StarSPIRE

StarSPIRE[7] is a visual text analytics tool. Analysts are afforded the ability to interact with a document set through direct manipulation of the visualization. The semantic meanings of those interactions are interpreted by the system as both updates to the layout that drives sensemaking and as updates to the relevance metrics that drives automatic document foraging. Underlying the interface of StarSPIRE is the multi-model, bidirectional pipeline shown in Figure 4.

In the forward direction of this pipeline, the collection of text documents (Data) is loaded into the system. A search algorithm serves as the Relevance Model, which judges the importance of each document and only displays the most important documents to the analyst via a relevance threshold. The Display Layout Model is a force-directed layout in which the resting distance of each edge corresponds to a similarity measure between each pair of documents. The output of this force-directed layout is displayed to the analyst and affords interactions such as moving document nodes, annotating documents, searching for terms, and linking document nodes.

Following an interaction, StarSPIRE interprets the semantic meaning of interaction to determine how to update the appropriate model parameters. These updates to the models are accomplished by an inverse of the computations used to generate the visualization. These inverse computations determine how the parameters of the models must be altered to generate the output that the analyst specified via the interaction. Once the inverse computations are complete, the analyst interest parameters are updated, which triggers the forward computations of each of the models to incrementally update the visualization.

Since the analyst directly manipulated the visualization in the given interaction, and because this interaction was translated into a complex update to model parameters, this interaction qualifies as a type of V2PI. The power of this interaction lies in that the interaction itself is simple, but the results of the interaction allow analysts to simultaneously forage for information (bringing new documents onto the screen) and synthesize existing information (spatializing documents based on an expressed similarity between existing documents) without having to adjust model parameters directly themselves. In this case, this enables the analysts to stay focused on conceptual-level analysis (spatial layout) of the documents, while the algorithms handle the detail-level parameters (entity weights)[8].

Andromeda

Andromeda[6] is a visual analytics tool designed for high-dimensional quantitative data analysis . As specified in Andromeda's pipeline[15] (Figure 5), V2PI allows analysts to directly manipulate points in a Weighted Multidimensional Scaling (WMDS)[16;17] projection of the data. This triggers an *inverse* WMDS computation. The forward WMDS computation uses the high-dimensional data and the dimension weights to produce low-dimensional coordinates. However, the inverse WMDS computation uses the low-dimensional coordinates (from the user's interaction) and high-dimensional data to calculate a set of dimension weights that best produce the desired projection. After generating new dimension weights from the inverse WMDS computation, a forward WMDS computation is then used to reflect the results of the user interaction. More formally:

$$coordinates = \text{WMDS}(data, weights)$$
$$weights' = \text{WMDS}^{-1}(data, coordinates')$$

In addition to V2PI, analysts can interact with the attribute weights directly through Parametric Interaction (PI). To do so, analysts manipulate sliders located in an accompanying panel, which changes the values of the model parameters directly. The simplicity of this interaction, although still useful, provides a stark contrast to the power of OLI despite the simplistic interaction of dragging points on the screen.

Through the use of V2PI, Andromeda allows analysts to perform sensemaking tasks by directly manipulating the visualization. Thus, Andromeda supports the sensemaking loop of the Sensemaking Process. Although Andromeda does not allow for the foraging of new observations in the same way that StarSPIRE does, Andromeda does also support the foraging loop of the Sensemaking Process by allowing analysts to forage for new relationships and patterns in the dataset. Therefore, Andromeda supports the Sensemaking Process and V2PI to the same degree as StarSPIRE, albeit in a slightly different manner.

Thus, the key to V2PI is not a pinning action as suggested by Sacha et al.[18], but rather a machine learning process in which the inverse computation calculates the proper model parameters on behalf of the analyst, based on the new positions of the dragged points. This enables the analyst to perform observation-centric sensemaking tasks, while foraging for new relationships in the data.

Additional Examples of Semantic Interaction

In addition to the StarSPIRE and Andromeda systems, a number of tools have been developed that make use of semantic interaction to afford analysts with interactive control of visualizations. We briefly highlight some of these tools and pipelines here.

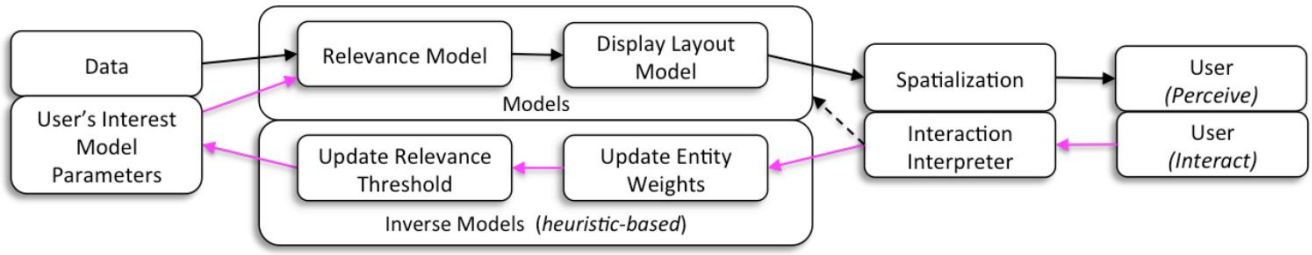

Figure 4. The bidirectional pipeline from StarSPIRE[7] allows analysts to interact directly with the visualization. Included in this pipeline are two models, the Relevance Model and the Display Layout model, which are each coupled with inverse computations (Update Relevance Threshold and Update Entity Weights). The models are chained together into an overall bidirectional structure that enables repeated interaction for sensemaking in a collection of documents.

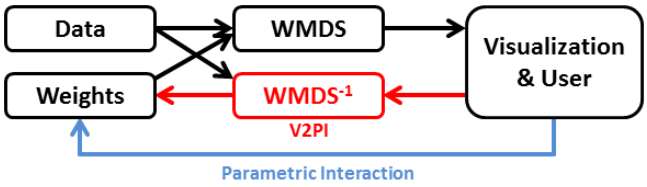

Figure 5. The Andromeda pipeline affords the analyst two methods to interact with the projection weights. If the analyst performs parametric interaction on the weights, those weights are modified directly and the execution path skips over the inverse WMDS computation. If the analyst performs V2PI by directly manipulating the projection, the inverse WMDS computation interprets the user interaction into an updated set of projection weights.

Intent Radar[22] introduces interactive intent modeling, allowing an analyst to provide feedback for search intents and thereby direct exploratory search. The analyst provides relevance feedback to the model by dragging or clicking keywords, increasing the relevance by moving the keyword closer to the center or decreasing it by moving it outward in the radar interface. This causes the system to compute new estimates for the analyst's current and future intents.

Dis-Function[19] presents a method for interactively learning distance functions for projections using a three-step process: (1) present the analyst with a projection, (2) allow the analyst to interact with the projection, and (3) update the projection based on the analyst feedback. The Dis-Function system also provides additional views alongside the scatterplot to better communicate the properties of the distance function to the analyst. The left pipeline from Figure 6 visualizes this distance function learning process.

Molchanov et al.[23] develop a system that uses cluster medians as control points that can be repositioned in the projection. When an analyst performs a repositioning action, the system interprets this action as an attempt to relocate the associated cluster or class of observations while not moving any other cluster or class. As it is often not possible to create a new projection that supports these precise conditions, the system computes a least-squares solution for a system of corresponding linear equations.

Paulovich et al.[20] propose a Piecewise Laplacian Projection as a multidimensional projection technique, with the process shown visually in the center pipeline of Figure 6. Here, control points are used to constrain the Laplacian system that governs the projection. When an analyst manipulates an observation in the projection, they induce a change in the Laplacian matrix associated with the neighborhood graph that includes the nearest control point to the manipulated observation. The update to the Laplacian matrix then drives an update to the projection.

Mamani et al.[21] propose a technique for modifying neighborhood structures in a projection, with the steps of this transformation shown in the right pipeline of Figure 6. A series of local affine mappings transform the projection in response to user-driven repositioning of observations.

These visual analytics tools and pipelines all share a number of common characteristics in their support of semantic interaction. Next, we begin to formalize the requirements of a pipeline that reflect these characteristics, providing a standardized way to think about and represent semantic interaction systems.

Characteristics of Semantic Interaction Tools

When comparing the characteristics of the above visual analytics tools, we note that there are several commonalities. Combined with the ideas from the Sensemaking Process, we define the following three properties as necessary for supporting semantic interaction in visual analytics tools:

1. A Series of Models: The pictorial representation of the Sensemaking Process (Figure 2) contains six boxes representing progressively more refined collections of information. These collections are connected via a pair of processes that allow for this transformation of information. Semantic interaction specifically calls for such a representation of information refinement into a visualization via a model, with interactions altering the parameters of that model. In Andromeda, we see this concept of a model reflected by its use of WMDS and inverse WMDS computations, which can be represented by a single model. StarSPIRE takes this concept a step further by using two models: the Relevance Model and the Display Layout Model. Thus, a series of models are necessary to enable this direct manipulation of the visualization.

2. Looping Sensemaking Interactions: As discussed previously, the Sensemaking Process defines pairs of processes that allow for information to be progressively transformed. These pairs of processes allow the transformation to go either top-down or bottom-up, implying that there is a concept of looping between these collections of information. This looping structure can be seen in StarSPIRE and Andromeda,

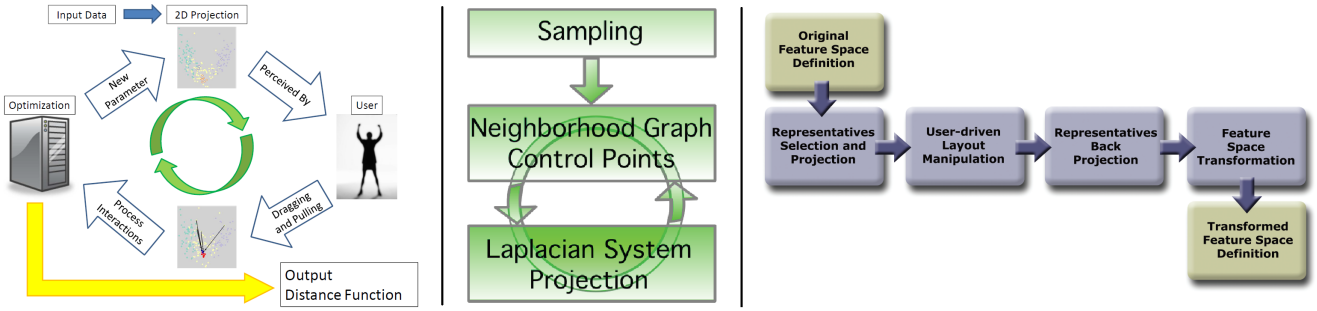

Figure 6. Pipelines from Dis-Function[19] (left), Piecewise Laplacian Projection[20] (center), Mamani et al[21] (right). Each of these pipelines reflects the implementation of a system that uses V2PI as an interaction method, despite the differing pipeline structures.

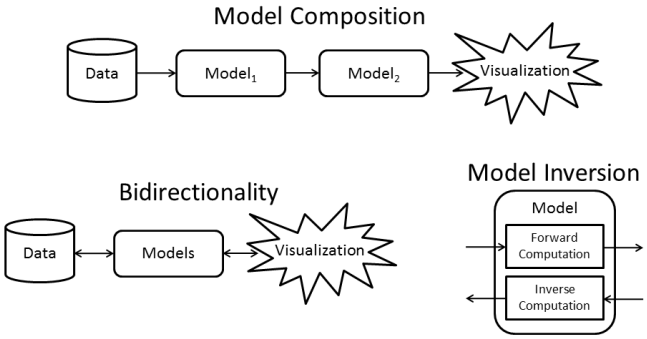

Figure 7. A representation of our three requirements for a new semantic interaction pipeline: Model Composition, Bidirectionality, and Model Inversion. Model Composition refers to the chaining of a series of models, each of which support one semantic computation. Bidirectionality allows user interactions to drive updates to the underlying models. Model Inversion refers to the pairs of a forward computation with an inverse computation. The inverse computation supports the translation of semantic interactions into manipulations of model parameters.

which each allow for models and their associated computations to be rerun iteratively based on interactions performed. It is not enough to use a series of models to visualize data; the visualization must be interactive and the analyst must be allowed to provide feedback to the underlying models.

3. Inverse Interactions: In order to loop between each step in the Sensemaking Process, each bottom-up process must have an associated top-down process that allows for feedback and learning. That is, each bottom-up process must have an inverse associated with it. In V2PI, this is represented by (1) using parameters θ to produce the visualization V, (2) allowing analysts to provide cognitive feedback by directly manipulating V and thereby producing V', and (3) recomputing θ to generate a new visualization. In Andromeda, this is clearly represented by the fact that the WMDS computation has an associated inverse WMDS computation to perform this inverse calculation based on user interactions. StarSPIRE has similar properties, as seen by the manner in which user interactions manipulate both the Relevance Model and Display Layout Model.

Pipeline Requirements

The three properties of visual analytics tools that use semantic interactions described in the previous section lead us to propose three corresponding requirements for a new visual analytics pipeline that enables representations of the complexity involved in semantic interaction:

1. Models should be composable, able to operate in sequence by taking as input the result of the previous model and sending as output the result of its computation to the next model. The idea of composable models is a well-addressed problem[7;11;18], although literature describing similar tools may use different terminology. As this is not a novel contribution, we will not focus on this point. Instead, we simply note that composable model structures support unknown numbers and combinations of models that can perform a variety of functions. As discussed previously, composable model structures are also seen in the Sensemaking Process to transition between multiple stages of the overall process. This is thus a necessary feature in a generalizable visual analytics pipeline that supports semantic interactions. The composable model structure can be seen in the upper pipeline in Figure 7.

2. The pipeline should support a looping structure through bidirectionality, allowing each user interaction to drive a backward and forward iteration through the pipeline. Based on our previous discussion of the usefulness of a looping structure, we suggest a requirement of a bidirectional structure, as shown in the lower-left pipeline in Figure 7. This requirement directly supports the foraging and sensemaking loops in the Sensemaking Process. With this bidirectional pipeline, user interactions drive updates to the underlying models.

3. Each model should comprise both a forward computation and an inverse computation, supporting the bidirectional structure. The need for bidirectionality combined with the more tightly-coupled loops that join each stage of the overall Sensemaking Process lead to a need for model inversion. That is, to properly interpret a semantic interaction, an inverse computation is needed to translate the semantic interaction within the visualization to manipulations of model parameters. Thus, a generalizable pipeline that supports these semantic interactions requires an inverse computation that accompanies the forward computations of each model. This inversion also keeps analysts' mental process tightly coupled to the data projection so that they do not have to leave their foraging or sensemaking loops in order to interact with the system. The model inversion structure is represented in the lower-right pipeline in Figure 7.

One important feature model inversion is the ability to short circuit the rest of the pipeline, which is a

key new feature of a multi-model pipeline not found in earlier definitions[7]. Short circuiting happens when the inverse computation of a model does not need to send the interaction any further down the pipeline. As an example, our Andromeda implementation (shown as a part of Figure 9 and discussed in the *Web Andromeda* subsection of the *Pipeline Implementations* section) enables a V2PI interaction. When this occurs, an inverse computation is triggered in its Similarity Model. However, since all the points are already visualized, there is no need to communicate with the Data again. Thus, instead of running the entire pipeline, we short circuit, executing the forward computations beginning at the Similarity Model to update the Visualization. This short circuiting is represented by the dotted upward arrow in the lower-right pipeline in Figure 7.

When evaluating traditional visual analytics models (e.g., Keim et al.[12]), we note that there is rarely a distinction between different models that may be used in the pipeline. Furthermore, while bidirectionality may be represented on some level, the manner in which the visual analytics pipeline handles this bidirectionality is not discussed or represented in detail. As a result, there is no representation of inverse computations. Thus, there is a need a new pipeline for visual analytics tools that better supports these requirements for semantic interaction.

Refining the Definition of Semantic Interaction with a New Pipeline

Given these requirements, we introduce a new generalized pipeline for semantic interaction-enabled visual analytics tools. Endert et al. introduced semantic interaction for visual analytics and defined the concept with seven broad principles[3]. With the introduction of this bidirectional visual analytics pipeline, we refine this notion of semantic interaction more specifically to refer to computations that make use of composable models, a bidirectional flow, and inverse computations. This refined definition for semantic interaction still supports the seven principles of semantic interaction defined by Endert et al. For example, we still support analysts' spatial cognition, use semantic interactions within the visual metaphor, and shield the analyst from the complex underlying models.

At a concrete level, we require properties of the pipeline to map back to the model composability, bidirectionality, and inversion requirements discussed previously. Having composable models means that each model individually should be self-contained and should have a standardized method for both receiving information from a previous model and passing information onward to the next model. In other words, these models should have a plug-and-play functionality. To support bidirectionality, each semantic interaction should trigger an series of inverse computations to update the models, which in turn triggers an entire iteration of the backward and forward directions of the pipeline. This bidirectionality and the inverse computations define how the system interprets and processes each semantic interaction.

Given the described requirements for a new semantic interaction pipeline, we specifically define the new pipeline to consist of Data, a series of Models, and the Visualization. This pipeline is shown in Figure 8. A brief description of these components is also included in Table 2, along with descriptions of the instantiations of these components in our example applications. Model composability is addressed through a series of Models. Each of these Models contains a forward computation and an inverse computation, supporting the inverse computation requirement. The bidirectionality of the overall pipeline is handled through transitions through these computations, using the forward computations in the forward direction and the inverse computations in the backward direction to loop through the entire pipeline in response to a semantic interaction. Thus, this proposed structure accurately captures the power and complexity of semantic interactions.

Using the Pipeline to Model Existing Semantic Interaction Systems

With this pipeline structure, we have the ability to model the behavior of the existing V2PI-enabled semantic interaction systems described in previous sections. Figure 9 shows a mapping between the pipelines provided in the original system descriptions (shown previously in Figures 4, 5, and 6) and our interpretation of the system behavior as modeled by our semantic interaction pipeline. From top to bottom in Figure 9:

Andromeda supports a scatterplot projection system that responds to analyst feedback through V2PI interaction, and also contains a method for updating the dimension weights directly (Parametric Interaction). Therefore, the V2PI portion of Andromeda can be modeled with a Projection Model in which the forward computation renders the projection and the inverse computation interprets and responds to analyst feedback. The Parametric Interaction features can be contained in an "Andromeda Model" that extends from the Projection Model. Our implementation of Web Andromeda using our pipeline, discussed later in the *Pipeline Applications* section, details these models to a much greater degree.

StarSPIRE requires both a Projection Model for the projection as well as a Relevance Model to handle the relevance computations. Because the projection depends upon the relevance values of the documents, the Relevance Model should be the first to run in the pipeline. The forward computation of the Relevance Model computes the relevance values of the documents, while the inverse computation learns and updates the relevance display threshold based on analyst feedback. The Projection Model projects the documents and responds to analyst feedback much like the Projection Model discussed for Andromeda, though using a force-directed method for projection instead of WMDS.

Dis-Function displays, among other views, a scatterplot of projected pairwise distances using Principal Component Analysis (PCA). An analyst is able to interact with that projection to provide incremental feedback, which in turn updates the projection. This behavior is quite similar to that of the Projection Model in Andromeda, although using PCA instead of WMDS. As such, we can model Dis-Function with a Projection Model that projects in the forward computation using PCA and responds to analyst feedback in the inverse computation.

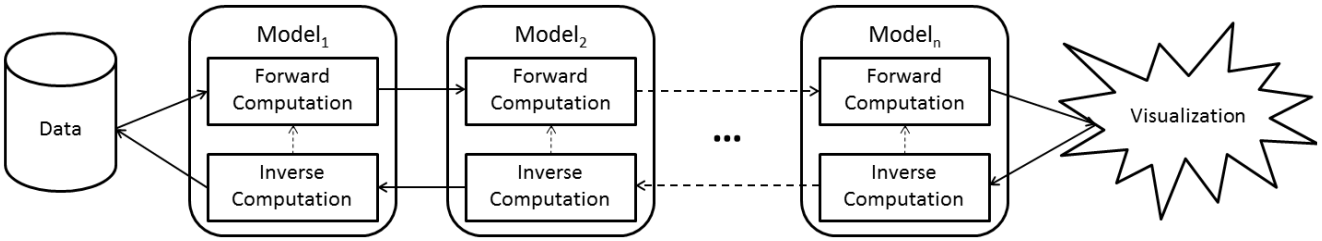

Figure 8. Our proposed semantic interaction pipeline, resulting from the combination of the three requirements shown in Figure 7. Model composability is shown through the chaining of a series of models horizontally in the pipeline. Bidirectionality results from the separated forward (top) and inverse (bottom) paths through the models. Model inversion is again shown through the pairing of a forward computation and an inverse computation in each of the models. This representation also shows short circuiting arrows that connect the inverse and forward computations in the Models. The result is an interactive system that transforms data into a visualization, and enables the manipulation of the visualization to better explore the data.

Paulovich et al. present a Piecewise Laplacian projection system in which samples are drawn from a full dataset, control points are created for each sample, and a neighborhood graph is constructed for the full dataset. As an analyst manipulates the projection, these control points and neighborhood graphs dynamically update. Using our pipeline, we can model this process using a Sampling Model to perform the sampling step and a Projection Model to perform the projection. In the Sampling Model, the forward computation performs the initial sampling step, while the inverse computation is unnecessary and can be implemented as an identity inverse (though a computation could be added here if the samples were updated based on analyst feedback). In the Projection Model, the forward computation creates the control points and projects the data using the Laplacian system, while the inverse computation interprets and responds to analyst feedback from manipulating the projection. Because this is the main loop within the system, the Projection Model can short-circuit, immediately updating the projection after the analyst feedback is received.

Mamani et al. propose a technique similar to that of Paulovich et al., though the precise mathematics of the projection differ (based on local affine mappings rather than Laplacian). Still, the basic process of sample first and project second remains the same in the forward computations. The process of responding to analyst feedback and then reprojecting incorporates a fresh set of samples considered by the interaction, therefore the full inverse computation process runs through all models rather than short-circuiting early in the Projection Model. Therefore, this pipeline is identical to the Paulovich et al. pipeline at the model level, though the mathematical details of the Sampling and Projection Models differ.

Concrete Pipeline Implementation

Here, we discuss the concrete components that we designed and implemented to demonstrate this new semantic interaction-enabled visual analytics pipeline. We discuss the goals of our implementation, and describe the modular components that combine to form the semantic interaction applications that are described in the next section.

Implementation Specification

To implement this new pipeline, we impose several specifications to support our own personal research goals. The following personal specifications are expanded upon through the rest of this section:

1. To allow for rapid prototyping within the pipeline, it must be easy to alter existing components or add new components.

2. To allow us to freely interchange components, each component of the pipeline must be self-contained. This enables us to easily create new pipelines, thus enabling experimenting with different visualizations, interaction techniques, models, and data types.

3. To allow for fast prototyping of different visualizations, the visualization itself must be separated from the rest of the pipeline. That is, the visualization only needs to be aware of the finalized data that will be visualized, and must also be able to communicate user interactions back to the pipeline.

4. To simplify development of the visualizations and as a method for sharing and distributing the visualization tools, the visualization itself must be a web interface, with the pipeline for the visualization running on a server that sends computed data to the visualization and listens for user interactions to process.

To facilitate the exploration of this design space, we wish to rapidly prototype each of our implementations. We chose to implement our pipelines in Python, since it is a suitable language for prototyping. Python is also widely used in data and text processing, with a vast array of available libraries and packages for doing so. As a result, Python enables us to use a wide array of tools already created while allowing for easy changes in the future. Thus, we can easily alter existing components of the pipeline or add new components as we choose.

We further designed each component of the pipeline to be independent and modular. As a result, none of the components of our pipeline are individually complex, but the independence of each component and communication protocol defined between them (which is discussed in more detail in the *Communication within the Pipeline* subsection) allows for semantic interactions and plug-and-play functionality with little effort.

In order to separate the visualization from the rest of the pipeline, we define an intermediate component, the Connector, that is responsible for handing the necessary

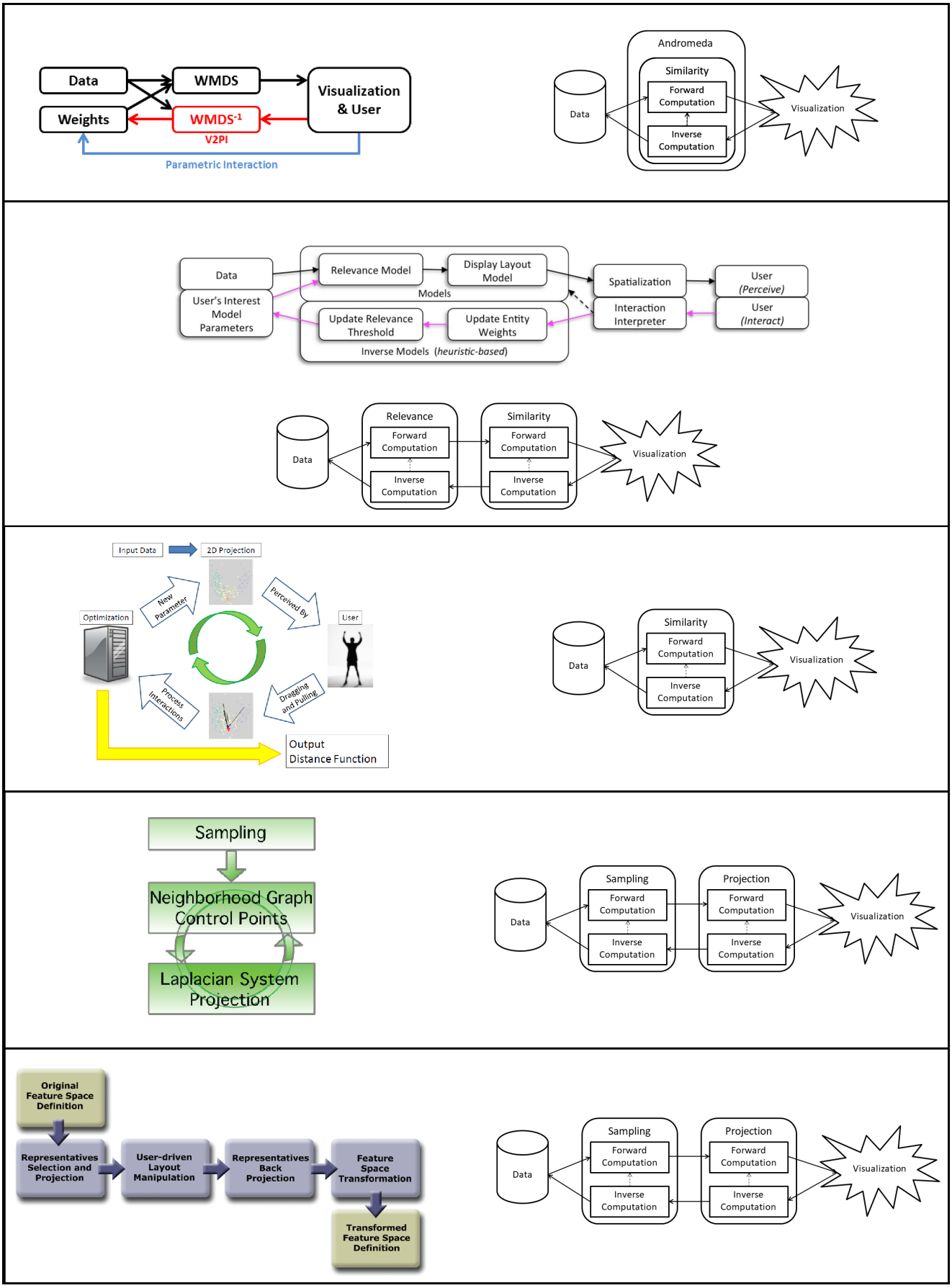

Figure 9. Using the proposed semantic interaction pipeline shown in Figure 8, we can now model the behavior of existing semantic interaction systems like Andromeda (Figure 5), StarSPIRE (Figure 4), and DisFunction, Piecewise Laplacian Projection, and Mamani et al. (Figure 6).

data to the visualization and communicating user interactions back to the rest of the pipeline. Defining our implementations in this manner allows us to create separate visualizations for each pipeline instance, which enables easy experimentations with different UI components, visual encodings, and semantic interactions. Finally, we made use of a web framework that enables us to create separate visualizations and associate URLs with them.

Combining these solutions together with our new semantic interaction pipeline led us to identify common structural components that are necessary within each implementation, and hence to define four key components of our pipelines: a Data Controller, a series of Models, a Connectorand a Visualization. Figure 10 illustrates these components and how they interconnect. Each of these components are further described in the following subsections, along with a description of how the components communicate with each other and how our pipeline is constructed.

Data Controller

The Data Controller serves as the main access point to the underlying data that is being visualized. Its key purpose is to serve as a method to retrieve the raw data and any possible metadata. This can enable analysts to view the raw data directly or allow the pipeline to pull additional data to process and visualize. A variety of Data Controllers can be developed to work with different types of data, enabling fast prototyping with different types of data by merely switching to a different Data Controller. For example, a pipeline that is designed to analyze numerical data can be changed to analyze textual data just through the creation of a new Data Controller. Thus, the remaining pipeline components would not have to be changed. Additionally, the Data Controller can perform any data preprocessing, such as data normalization, or leave this step for a Model to perform.

Models

Between the Data Controller and the Connector is at least one Model. However, multiple models can be chained together in a sequence to enable more complex visualizations and/or interactions. Each model is comprised of a forward computation and an inverse computation. The forward computation uses a set of parameters to manipulate the data, which either originates from the Data Controller or from a previous Model in the pipeline. That is, each Model's forward computation operates in sequence, defining a specific method for how the data is processed as it works its way from the Data Controller to the Visualization.

In contrast, the inverse computation of each Model interprets the analyst's interactions and updates the parameters that the Model's forward computation uses accordingly. To do so, the inverse computations operate in the opposite order of the forward computations, starting with the inverse computation of the Model closest to the Visualization and working towards the Data Controller. Thus, the sequence of inverse computations determine how the analyst's interaction is interpreted. These updates to the Model parameters are then used by the forward computations of each of the models, ultimately reflected in an updated Visualization. In the case of our pipeline implementations, we suppose a wide variety of inverse computations, include precise mathematical inverses[6], heuristic inverses[27], probabilistic inverses[4], and even identity function inverses in which no computation occurs (for example, see our discussion of the Term Frequency inverse computation in the Elasticsearch pipeline in the next section).

Connector

With the goal of putting our Visualizations on the Web, we designed the Connector with two distinct components: the Pipeline Connector and the Visualization Connector. The Visualization Connector is a Node.js web server, listening to messages from the Visualization and sending messages to the Visualization. Only the Visualization Connector has direct communication with the Visualization itself. The details of the implementation of the Visualization Connector are discussed in the *Web Framework* subsection.

In contrast, the Pipeline Connector is part of the Python pipeline. Instead of communicating with the Visualization, it communicates with the Visualization Connector. When the Visualization Connector receives a message from the Visualization that necessitates a change in the pipeline, it sends a message to the Pipeline Connector. These messages between the Visualization Connector and Pipeline Connector come in three forms:

- The *Update* message signals that the analyst has performed some interaction that warrants an execution of the pipeline, starting with the inverse computation of the Model closest to the Pipeline Connector. An *Update* message may make its way through the entire pipeline back to the Data Controller, or it may get short circuited by one of the Models along the way.
- A *Get* message is sent in response to an analyst's request for more information about an observation. These are sent to the Data Controller for retrieving raw data or metadata directly, which is then returned to the Pipeline Connector to forward to the Visualization Connector and, eventually, the Visualization.
- A *Reset* message that signals all the Models and the Data Controller of the pipeline must reset back to their initial state.

A more detailed description of inter-pipeline communication and the role of the Connector is provided in the *Communication within the Pipeline* subsection, after all of the components have been introduced and the web framework has been more thoroughly discussed.

Visualization

Each Visualization we have created is a D3[28] interface that communicates with the pipeline through the socket.io JavaScript library[25]. To communicate with the pipeline appropriately, the Visualization first creates all necessary visualization components and defines when to send socket.io messages to the pipeline or when to respond to socket.io callbacks. The Visualization sends messages when an analyst performs an interaction that requires a response from the pipeline and defines callbacks for how to respond to information coming from the pipeline. For example,

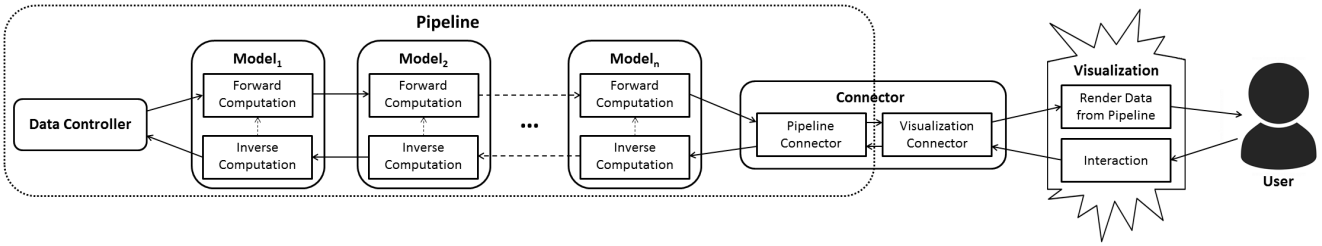

Figure 10. An instance of our concrete implementation of the semantic interaction pipeline shown in Figure 8 consists of a Data Controller, series of Models, and Connector. The Visualization is a separate entity from the pipeline, communicating with the pipeline through the Connector. The solid arrows between the models indicate the forward and inverse flow in the pipeline, while dotted arrows within the Models indicate the ability to short circuit. The features of model composition, pipeline bidirectionality, and model inversion are still apparent in this concrete implementation of the pipeline.

Component	Description
Connector	Serves to connect the pipeline Models to the Visualization so that they can operate independently. It is comprised of two pieces: a Pipeline Connector written in Python that interfaces with the Pipeline, and a Visualization Connector that interfaces with the Visualization as the Node.js[24] server using socket.io[25]. These pieces use ZeroRPC[26] to pass messages between each other.
Data	The content that acts as input to the pipeline to generate the Visualization.
Data Controller	The pipeline component that accesses and interfaces with the underlying data that is being visualized. This is not a "Controller" in the MVC software architectural pattern as it does not control the messages that are passed through the pipeline. Rather, it is only responsible for retrieving raw data as requested by a Model or by a *Get* message from the Pipeline Connector. Data Controllers that are implemented in our example applications include: • *CSV Data Controller:* A Data Controller that reads and parses a CSV file. • *ElasticSearch Data Controller:* A Data Controller that connects to an ElasticSearch text search engine for dynamic datasets.
Model	A single unit of mathematical manipulation that is comprised of a forward computation and an inverse computation. The forward computation is used to manipulate the data on its way to the Visualization, while the inverse computation defines how an interaction in the Visualization is to be interpreted. Models that are implemented in our example applications include: • *Relevance Model:* Acts as a filter to determine which observations should be included in the Visualization. • *Similarity Model:* Creates low-dimensional coordinates for high-dimensional observations using a "proximity ≈ similarity" metaphor. – *Andromeda Model:* An extension of the Similarity Model that also supports parametric interaction. • *Term Frequency Model:* Performs entity extraction and computes entity term frequencies.
Pipeline	The Python pipeline components, including the Data Controller, Models, and Pipeline Connector.
Visualization	The visual output of the Pipeline that supports user interaction.

Table 2. Description of the components in our semantic interaction pipeline. To differentiate between a pipeline component and a general term defined in Table 1, we capitalize the pipeline components (e.g. "Model" vs "model")

an *Update* message is sent to the pipeline when an analyst performs V2PI in our Web Andromeda interface (as described in the *Pipeline Implementations* section). The pipeline processes this message and returns data to the Visualization, which is handled by a socket.io callback that uses the incoming data to move, create, or delete DOM objects in the webpage.

Web Framework

We also want to implement the Visualizations on a web framework, which allows for rapid development and distribution of these Visualizations. Additionally, we saw an opportunity to move visualization research to the Web, using mature web-based graphical frameworks that enable complex visualizations in the browser[28–30]. This mirrors the recent trend of transitioning from native applications to web applications[31;32]. Utilizing these frameworks requires a server to handle the Visualization itself. Combined with

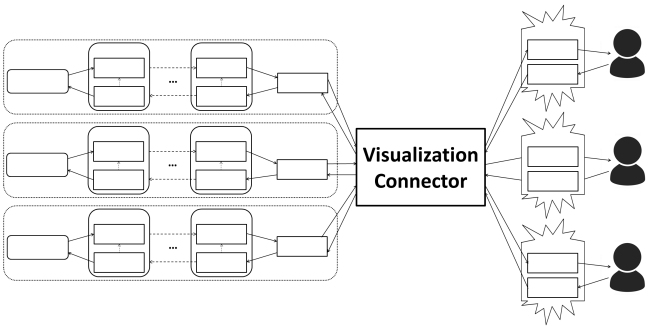

Figure 11. The Visualization Connector can mediate communication between multiple pipelines and Visualization clients. A single Visualization can be mirrored across multiple clients.

the communication requirements for the Pipeline Connector, this resulted in the creation of the Visualization Connector to act as the server and handle communication between the Visualization and the rest of the pipeline.

The Visualization Connector is implemented in Node.js[24] and handles web components such as managing analyst sessions as well as communicating with both the Visualization and the Pipeline Connector. Communication with the Pipeline Connector is accomplished using ZeroRPC[26], but communication with the Visualization is accomplished via WebSockets[30].[†] Using WebSockets has additional benefits such as allowing Visualizations to be mirrored to multiple analysts simultaneously. This enables new areas of research on multiple analysts collaborating within a single Visualization to be addressed[33].

Figure 11 illustrates how the Visualization Connector allows for multiple analysts to be connected to different instances of the pipeline. To accomplish this, each pipeline instance is tied to a single ZeroRPC connection within the Visualization Controller. Additionally, each analyst is also connected to the Visualization Connector via a single WebSockets session. Within the Visualization Connector, each WebSockets session is associated with a particular Visualization type, then paired with a particular pipeline instance. This enables a many-to-many relationship between user sessions and pipeline instances.

The flexibility of our implemented pipelines demonstrates how adaptable our new visual analytics pipeline is. This adaptability provides many benefits towards creating visual analytics tools. As seen in the prototypes in the next section, we are able to create many different prototypes by swapping out individual components. This makes rapid prototyping simple and provides us with the ability to quickly implement new ideas in the pipeline.

Communication within the Pipeline

Due to the separation between the Visualization and the Models, the Connector is responsible for handling messages between the two. For the Pipeline Connector, this communication is accomplished using ZeroRPC[26]. ZeroRPC is a Remote Procedure Call implementation of ZeroMQ[34], an asynchronous messaging library for distributed applications. The Pipeline Connector creates a ZeroRPC server with RPC bindings for the *Update*, *Get*, and *Reset* messages. The Visualization Connector then connects to the Pipeline Connector as a ZeroRPC client, establishing communication between the two.

Separately, the Visualization Connector establishes communication with the Visualization by acting as a Node.js[24] server. The Visualization then connects as a client to this server, using socket.io[25] to pass messages between itself and the Visualization Controller. These messages mirror the *Update*, *Get*, and *Reset* messages used in the RPC bindings in the Pipeline Connector, making translating the socket.io messages to RPC messages easy.

Thus, updates to the Models are driven by interaction with the Visualization. This results in the Visualization sending socket.io messages to the Visualization Connector when the analyst performs an interaction the requires an update or response from the Python-implemented portions of the pipeline (represented by the outer "Pipeline" box in Figure 10) . The Visualization Connector then translates these socket.io messages to RPC messages to send to the Pipeline Connector. The Pipeline Connector interprets this message to either get information directly from the Data Controller or to forward the message to the Model closest to the Pipeline Connector in the Pipeline. If the message is passed to a Model, such as in the case of semantic interactions like V2PI, the message progresses clockwise around the Models shown in Figure 10. Each Model's inverse computation then determines how the interaction should be interpreted, deciding to short-circuit when no other Models are needed to process the interaction.

After all necessary inverse computations have been completed, the forward computations of each of the Models begin to run in the opposite order as the inverse computations were performed. These forward computations may spatialize the data, filter based on relevance, assign observations to clusters, or any other number of computations necessary to generate the desired Visualization.

Within the Pipeline itself, communication is accomplished using a global JSON-like object that is shared among these pipeline components. This communication object is initialized when a new Pipeline is created and stores all aspects of the data as it is process for the Visualization. If two Models must share data or modify the same data, this can be accomplished by having both Models operate on the same elements, or keys, within the communication object.

To ensure that the different components of the pipeline can communicate with each other appropriately, each Model specifies requirements for its forward and inverse computations. These requirements are specified as keys within the communication object, and then are checked against the output of the previous Model (or the Data Controller in the case of the first Model in the pipeline). If the input requirements of all computations are met, then the pipeline is valid. Checking for a valid pipeline prevents unwanted behavior caused by incorrectly constructed Models.

Figure 12 illustrates how communication in the Pipeline works using a single Model as an example. The forward

[†]Further details on how we use ZeroRPC and WebSockets are provided in the next subsection.

computation of this WMDS model is given the set of dimension weights that is applied to a dataset of animals, as well as the source data (the animals and their attributes). The Model's forward computation calculates a set of low-dimensional coordinates to position those animals in the projection. The inverse computation is given the analyst-modified screen coordinates of a set of animals, as well as the same animals and attributes source data, and computes a new set of dimension weights to apply to the projection. This allows each Model to contain only the logic required for transforming the data, and not the logic necessary to communicate with other pieces of the pipeline.

When the Pipeline has completed processing the message sent by the Connector, the Pipeline Connector retrieves portions of the communication object that the Visualization needs to know about (e.g. the final coordinates of the datapoints) and passes them to the Visualization Connector. In turn, this information is forwarded back to the Visualization, which uses the new data to update itself. Thus, the analyst is automatically provided an updated visualization based on their interaction.

Adding a New Feature to an Existing Pipeline

Adding a new feature to an existing pipeline requires modification to the pipeline, but the location of these modifications is dependent upon the complexity of introducing the new feature. For example, introducing a new color mapping to the Visualization requires only an update to the Visualization itself. In contrast, adding a new semantic interaction to the Visualization requires modification to the Visualization to support that interaction as well as to the Model will interpret that interaction and making the necessary changes to the model parameters. If no Model exists that naturally addresses the new interaction, an entirely new Model may be introduced. Finally, adding support for a different data type may require changes at the Data Controller level, and Model level, and the Visualization Level of the pipeline in order to retrieve, process, and display the new data.

Visualization Variations The role of the Visualization is to accept data from the pipeline and render it, and also to send messages to the pipeline in response to user interactions. To add a new feature, some modification needs to be made to each of these actions. After the forward computations of the pipeline have fired and data has been passed to the Visualization, some encoding must be implemented to map the data to features of the Visualization.

When developing a new Visualization or modifying an existing Visualization, a programmer must therefore:

- Specify a visual encoding for the data coming from the pipeline.
- Implement some interaction on that visual encoding.
- Format a message to fire down the pipeline in response to an interaction.

Model Modifications As shown in Figure 12, each Model receives messages and data into both the forward and inverse computations. Messages in either direction can be passed from another Model, or can originate with the Data Controller in the case of the forward computation or the Connector in the case of the inverse computation.

When a new feature has been added to a Visualization, the programmer must either modify an existing Model or create a new Model to respond to user interactions on that feature. In either case, implementation must be created or modified on a forward computation and/or an inverse computation.

More specifically, in developing a new Model or modifying an existing Model a programmer must:

- Define the forward computation that accepts data from the Data Controller or a previous Model and sends data to the Connector or the next Model.
- Define the inverse computation that receives an interaction message from the Connector or a previous Model.
- Determine under which conditions the inverse computation should short circuit vs. which conditions the inverse computation should continue sending the message and JSON-like object along the pipeline to another Model or to the Data Controller.
- Specify how the new or modified Model will modify the JSON-like object before passing it along to the next Model in the sequence both directions.

Data Controller Differences If the new feature requires support for an additional format of data, then a new Data Controller may need to be introduced. The Data Controller must retrieve data from an external location and reformat it to be interpreted by the rest of the pipeline. This may not always be possible, and so a modification to the Data Controller may also require changes to the Models and Visualization. In general, developing a new Data Controller or modifying an existing Data Controller requires the following programmer tasks:

- Define a method of retrieving the data from the file, database, or other source.
- Define a method for structuring and inserting the data into the communication object in a way such that the existing Models and Visualization can interpret the new data.
- If the previous step is impossible due to the scale of the change, alter the Models and Visualization as necessary to support the new data.

Pipeline Applications

With this pipeline, the interesting research challenges are shifted to discovering the mathematical techniques for investigating various combinations of data types, computations, visual representations, and user interactions, rather than creating a system for doing so. In this section, we illustrate five visual analytics applications that have been developed using this visual analytics pipeline. Our goal with these applications is to further explore the design space of visual analytics tools that enable semantic interactions using a variety of data types, models, interaction techniques, and

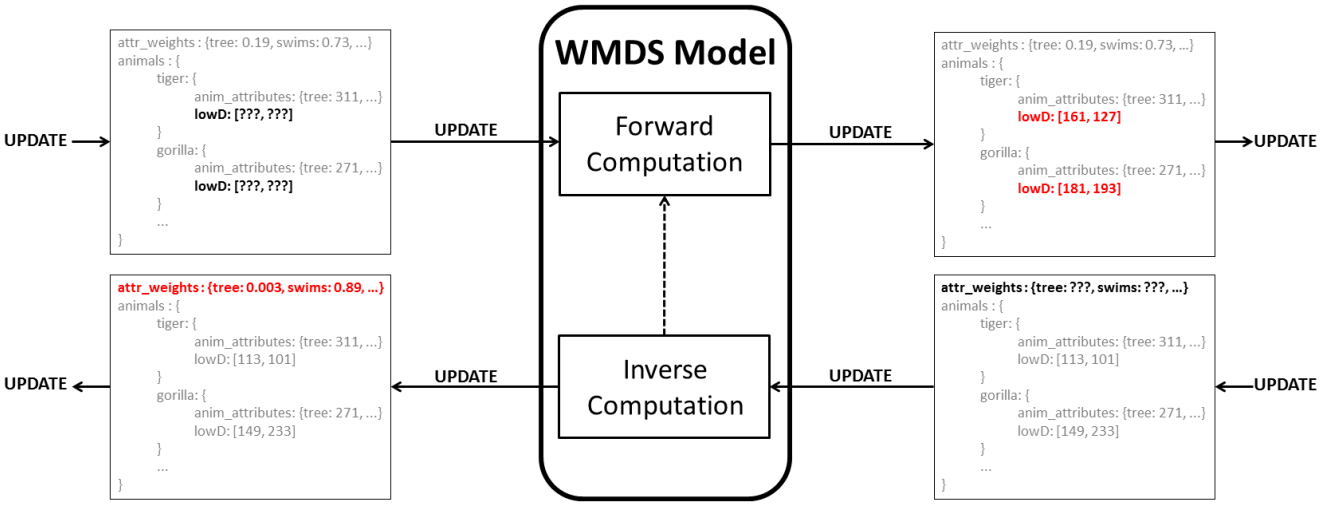

Figure 12. Forward and inverse computation examples detailing how a Model can modify the JSON-like object passed through the pipeline, in this case showing how the WMDS computation in our Similarity Model modifies the projection of a dataset of animals[35] that we also show in our Web Andromeda use case. In the forward computation, the data blob contains both the set of global weights applied to the projection and the collection of animals with their associated attributes. After running the forward computation, the MDS algorithm has computed a set of low-dimensional coordinates to place the animals in the projection. In the inverse computation, the data blob still contains the collection of animals with their associated attributes, but now includes some analyst-modified low-dimensional coordinates that represents the requested relationships in the data. After running the inverse computation, the MDS^{-1} algorithm has computed a new set of global weights to apply to the projection on the next iteration through the forward pipeline.

visualizations. These applications handle different types of data (numerical and text), and are constructed from two Data Controllers and four Models in various combinations. Each of the prototypes are discussed in the following format:

- **Motivation:** We begin by motivating the creation of the application, describing why such a tool is useful and what we could learn from it.
- **Pipeline:** We discuss the components required for each model, illustrating the flexibility and reusability of the pipeline components to create new applications.
- **Use Case:** We provide a use case for each of the applications to demonstrate how semantic interactions are afforded in each.

We focus these descriptions on the semantic interaction components of each of the applications. The applications themselves have further capabilities that may also be mentioned when appropriate (e.g., viewing raw data).

Web Andromeda

Motivation The first pipeline demonstrated is a web-enabled version of Andromeda, a visual analytics tool discussed earlier in this paper. We will refer to our implementation as "Web Andromeda" to differentiate it from the original Andromeda created by Self et al[15]. Due to the simplicity of its single-model pipeline, we use Web Andromeda as a baseline to evaluate the differences between our various pipeline implementations.

Pipeline The pipeline for Web Andromeda is shown in Figure 13. As this is our first implementation, we introduce the CSV Data Controller, the Similarity Model, and the Andromeda Model here.

CSV Data Controller: Web Andromeda uses data uploaded from a CSV file, which is z-score normalized before being handed to the Similarity Model. For the initial WMDS projection, all attributes are weighted equally ($1/p$). This Data Controller responds to *Get* messages from the Pipeline Connector, and sends the raw data associated with the requested observation directly back to the Pipeline Connector.

Similarity Model: The role of the Similarity Model is to spatialize documents according to their similarity. This is done using Weighted Multidimensional Scaling on the high-dimensional data passed down the pipeline. The Similarity Model stores a set of dimension weights, one for each dimension in the high-dimensional data. The forward computation uses these weights to project the high-dimensional data to 2D. This dimension reduction is performed by optimizing the location of each point in the low-dimensional space so that it minimizes a stress function between all pairs of points. Stress in this case is defined as the difference between the distance of two points in high-dimensional space and in low-dimensional space. This optimization is specified as:

$$d = \underset{d_1,\ldots,d_n}{\arg\min} \sum_{i=1}^{n-1} \sum_{j>i}^{n} \left(dist_L(d_i,d_j) - dist_H(W,D_i,D_j)\right)^2 \quad (1)$$

This computation is distance function agnostic; any distance function can be used in the model. Analysts can directly manipulate the WMDS projection of the observations by clicking and dragging nodes in the observation panel (V2PI). Once an analyst has moved their desired nodes, clicking "Update Layout" triggers the inverse WMDS computation, which seeks to optimize the weights to best reflect weighted high-dimensional distances in the user-modified projection. This optimization, described by Self et

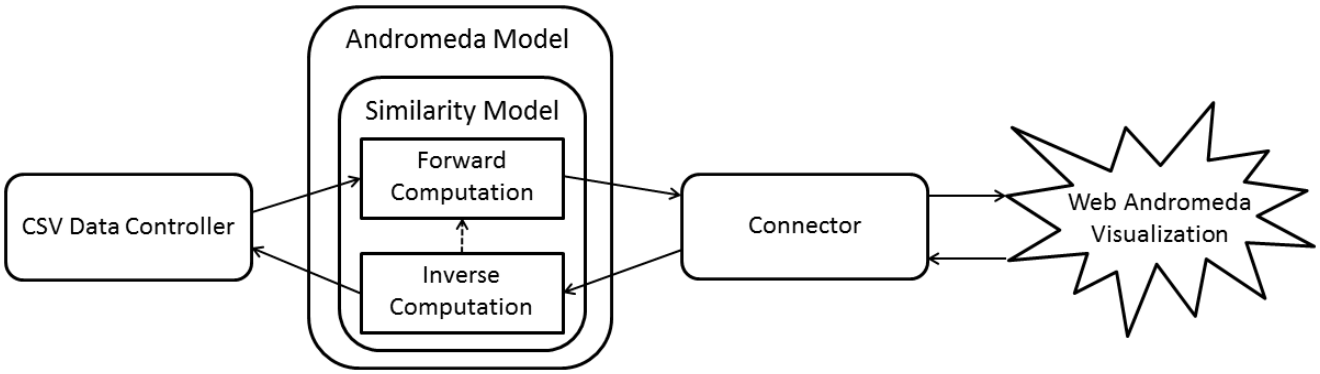

Figure 13. The pipeline for Web Andromeda includes a CSV Data Controller, an Andromeda Model (which inherits from a Similarity Model), a Connector, and the Visualization. This pipeline is structurally identical to the pipeline for Andromeda shown in Figure 5. From that pipeline, the WMDS and WMDS^{-1} computations have been combined into the Similarity Model, Parametric Interaction is handled by the Andromeda Model subclass (which inherits from the Similarity Model), the Data and Weights are loaded by the CSV Data Controller and maintained in the JSON-like object, and the Visualization renders the projection (with the Connector to enable communication to the Visualization).

al.[6], is specified as:

$$W = \underset{W_1,\ldots,W_p}{\arg\min} \sum_{i=1}^{n-1} \sum_{j>i}^{n} \left(dist_L(d_i^*, d_j^*) - dist_H(W, D_i, D_j) \right)^2 \quad (2)$$

This new set of weights is then used on the next forward projection of the data. When this interaction occurs, the Similarity Model short circuits the pipeline to immediately update the visualization based on the new similarity weights.

Andromeda Model: The Andromeda Model inherits from the Similarity Model and handles Parametric Interactions. When the analyst adjusts an attribute slider, the weight for that attribute is directly altered. Upon doing so, Equation 1 is recomputed, and the nodes in the observation panel transition to their new locations.

Visualization: As seen in Figure 14, Web Andromeda is a web based interface that is designed using D3[28]. This interface consists of two interactive panels: one for a WMDS projection of the observations of the high-dimensional data, and one for the sliders that represent the dimension weights used in WMDS. Analysts can specify how important an individual attribute is to them by interacting with the sliders. This interaction is the same PI described by Self et al.[6] and is interpreted as a desired weight for that attribute in the WMDS algorithm.

Analysts can directly interact with the projected observations as a form of V2PI. Analysts can drag nodes within the WMDS projection and use the "Update Layout" button to trigger the inverse model computation. When the computation is complete, the visualization updates to display the new WMDS projection, and the attribute sliders are also updated to reflect the learned dimension weights. Also, by hovering or clicking on a node in the WMDS projection, points appear along the attribute sliders to give analysts an idea of what attribute values are associated with the given observation. This interaction is exemplified in Figure 14-c.

Messages: The most important messages that are sent between the Visualization and the Connector are those that communicate the user interactions of PI and V2PI. Both of these interactions are communicated using an *Update* message. For PI, the *Update* message parameters indicate which attribute was manipulated. The Pipeline Connector forwards this message to the Similarity Model, which directly changes the parameters used in its forward computation to reflect this interaction; no inverse computation is needed for this interaction. For V2PI, the *Update* message specifies the coordinates of the moved nodes. The Pipeline Connector forwards this message to the Similarity Model, which changes the parameters used in its forward computation via its inverse computation. After either of these interactions, the Similarity Model's forward computation is rerun, with the new low-dimensional coordinates and dimension weights sent back through the Connector to be reflected in the Visualization.

Another interaction is hovering or clicking on a node in the visualization. This triggers a *Get* message sent directly to the Data Controller to obtain the raw data associated with the given node. This data is then passed back directly to the Pipeline Controller and is then forwarded to the Visualization Controller to be visualized.

Use Case In Web Andromeda, semantic interaction comes solely in the form of V2PI. Here, we demonstrate how V2PI can be used to gain new knowledge from an animal dataset[35].

After initially projecting the data (Figure 14-a), analysts can interact with the projection to learn more about the animals within the dataset. Perhaps the analyst is interested in learning more about what separates cats from dogs (a sensemaking task) and, based on how this difference is defined, what other animals can be considered similar to cats and dogs (a foraging task). After dragging cats to one side of the Visualization and dogs to another, (Figure 14-b), the analyst can click the "Update Layout" button to trigger an inverse computation in the Similarity Model. As a result of this interaction, the system determines that the "Claws" attribute best describes the difference between cats and dogs. This answers the analyst's sensemaking-related question. With the high weight of this attribute, the entire dataset is reprojected using the forward direction of the pipeline (Figure 14-c). From this new projection of the dataset, the analyst can see that animals such as Raccoon, Otter, Mole, and Beaver are similar to dogs and animals such as Grizzly Bear, Polar Bear, and Bat are similar to cats. Thus, the V2PI interaction answers the analyst's foraging-related question

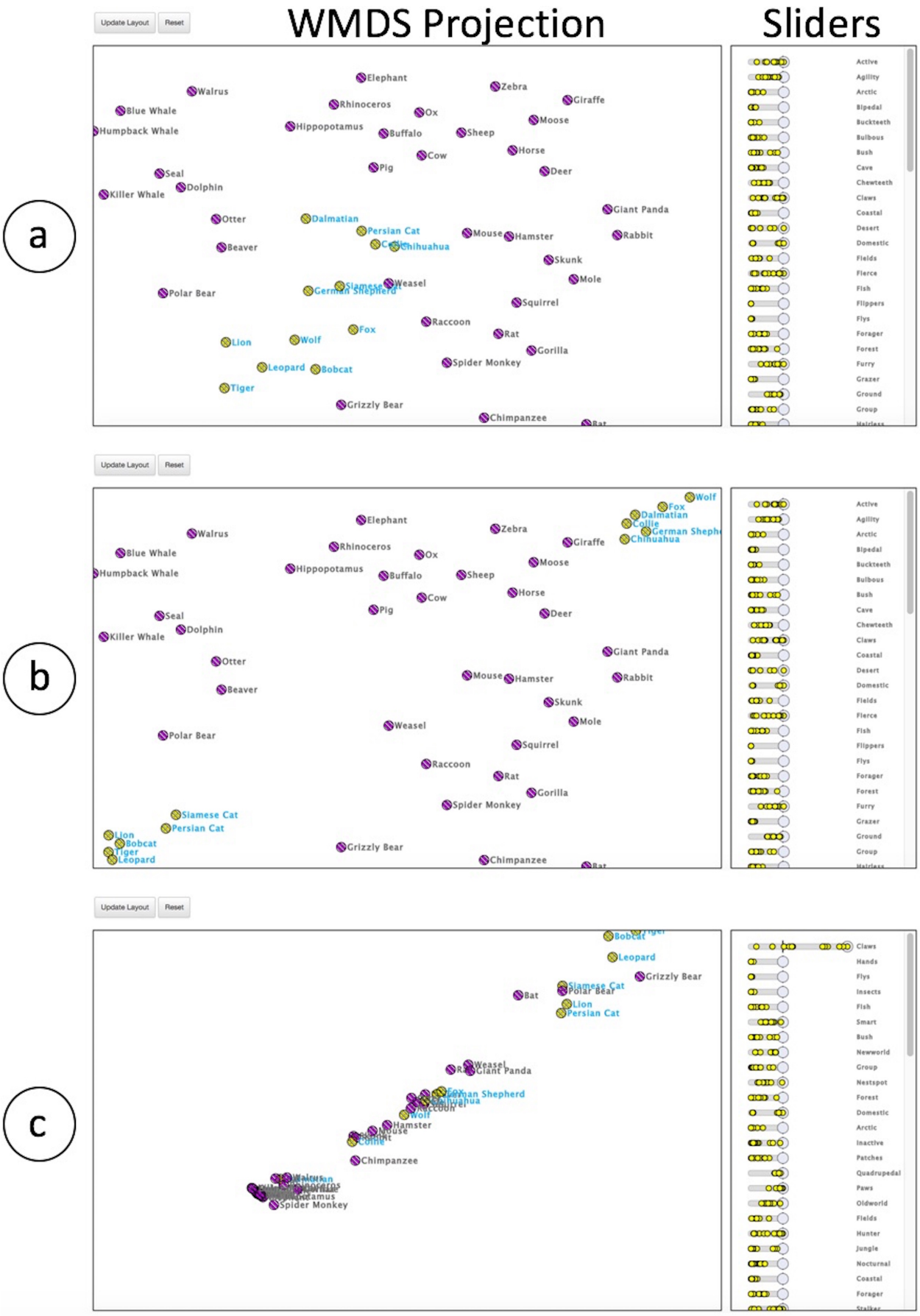

Figure 14. In Web Andromeda, the analyst can use V2PI to perform sensemaking tasks. After the initial projection **(a)**, the analyst can move points **(b)** and click the "Update Layout" button to trigger this inverse computation in the Similarity Model. After determining the weights of the attributes that best describe the pairwise distances between the moved points, the entire dataset is reprojected **(c)**. From this new projection of the data, the analyst can perform both foraging and sensemaking tasks.

as well, demonstrating the use of semantic interaction in sensemaking tasks with visual analytics tools.

Cosmos

Motivation The Cosmos pipeline was created to explore how the Similarity Model from the previous Web Andromeda subsection might be used for text analytics. To accomplish this, we combined aspects of the relevance-based retrieval model from StarSPIRE[7] with the Similarity Model. With this pair of models, analysts can query for new documents, view the raw text associated with a document, manipulate a document's relevance, and directly manipulate the visualization of the documents themselves. This V2PI interaction is represented in the transition between panels c and d in Figure 16.

In this new tool, we also introduce a *relevance* measure. By visually representing how *relevant* a document is to an analyst, we can also allow analysts to interact with this representation. If a document is considered more relevant, then other similar documents should also be considered more relevant and all these documents should be placed closer together in the Visualization. An example of this effect is shown in the transition from panel a to panel b in Figure 16. In this prototype, we chose to enable this interaction through a Relevance Slider.

Pipeline The Cosmos pipeline is shown in Figure 15. Note that this pipeline is similar to the Web Andromeda pipeline. The differences between the two pipelines are reflected in alteration to the CSV Data Controller and the addition of the Relevance Model. Thus, this pipeline demonstrates the plug-and-play functionality of our pipeline. In this section, we discuss the modified CSV Data Controller and the new Relevance Model introduced in the Cosmos pipeline. We also briefly discuss the Cosmos visualization and the semantic interactions supported by this pipeline.

CSV Data Controller: For this pipeline, we modified the CSV Data Controller from Web Andromeda to work with text documents. To do so, we first assume that the entire document set exists as a set of flat files on the server. We also assume that the CSV file that is used contains the TF-IDF values for entities extracted from the document set. On load, this Data Controller reads the data from the specified CSV file, preprocesses the data in the same manner discussed in Web Andromeda's CSV Data Controller, and sends the data along the pipeline to the Models. Additionally, this Data Controller adds references to the flat files on the server, necessary for some defined interactions. Aside from these changes, this Data Controller works in the same fashion as Web Andromeda's Data Controller.

Relevance Model: We drew inspiration from StarSPIRE to create our Relevance Model. The Relevance Model uses the same set of dimension weights that the Similarity Model does, but in a different manner. In the forward computation, this model computes the relevance of a document given a set of dimension weights as a linear combination of those weights and the document's TF-IDF values. This is represented by the equation:

$$r_i = D_i^T W, \tag{3}$$

This simple relevance calculation combined with a threshold determines which documents are passed on to the Similarity Model. That is, the Relevance Model can act as a filter that determines which documents are visualized.

The Relevance Model also has an inverse computation. An analyst can specify that they wish a document to have a certain relevance, and this inverse step determines what set of weights would give that document the desired relevance. Say an analyst changes the relevance of document i (i.e., D_i) from $(r_i)_{old}$ to $(r_i)_{new}$. The new dimension weights resulting from this interaction would be calculated using:

$$(W_i)_{new} = (W_i)_{old} + D_i \frac{((r_i)_{old} - (r_i)_{new})}{D_i^T D_i}. \tag{4}$$

Intuitively, this equation rescales the weight vector by another vector proportional to the feature vector for the document whose relevance is being changed. Following this determination, the forward computation is used to compute the relevance for all displayed documents.

The Relevance Model is also responsible for querying for new documents to display. Using the dimension weights, the Relevance Model finds the top n most relevant documents that are above the relevance threshold. This ensures that only highly relevant documents are displayed while also guaranteeing that the analyst will not be overwhelmed by too many documents appearing in the Visualization at once. However, this querying only occurs if the interaction dictates that a query is necessary.

Visualization: As shown in Figure 16, the Cosmos Visualization remains a web interface with two panels: one for an interactive WMDS projection of the documents (represented as nodes), and one that displays the details for a single document. This second panel takes the place of the attribute sliders in Web Andromeda. Unlike Web Andromeda, the WMDS panel is initially empty, requiring the analyst to search for a term. After documents nodes are placed on the screen, their relevance calculations are mapped to the sizes of the nodes. The analyst can then use the interactions described below to manipulate the Visualization.

Although V2PI and double-clicking document nodes occur directly within the WMDS projection, double-clicking a node populates the panel to the right of this projection with information specific to that document. This includes the label of a document node as well as the raw text of a document and associated notes. The analyst also has the ability to delete a document node by clicking a button on this panel.

Messages: A number of interactions are supported in Cosmos. Here, we address a subset of three interactions: (1) V2PI, (2) an analyst's query, and (3) manipulating the Relevance Slider associated with a particular document.

1. **V2PI**: As in Web Andromeda, V2PI is communicated to the Similarity Model via an *Update* message, which triggers the inverse WMDS algorithm. The results of this are provided to the Relevance Model to perform implicit querying on behalf of the analyst. The relevance of existing documents and any new documents are recalculated via the Relevance Model's forward computation. These documents are then sent to the Similarity Model to recompute appropriate low-dimensional coordinates.

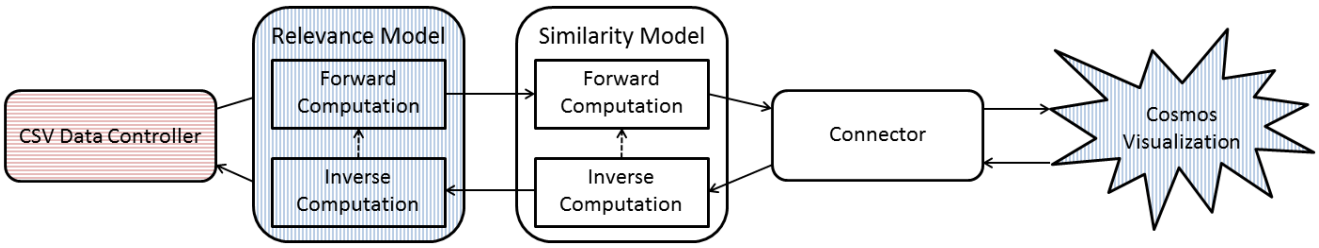

Figure 15. To transform the Web Andromeda pipeline into Cosmos, we introduce a Relevance Model, modify the CSV Data Controller, and create a new Visualization. The Relevance and Similarity models each handle a different component of generating the visualization, and are chained together in sequence, following our Model Composition requirement. The shaded, vertically-striped blocks represent entirely new components, while the shaded, horizontally-striped blocks represent components that previously existed but were modified from the Web Andromeda implementation.

2. **Analyst's Query**: An *Update* message containing the search terms is sent to the Relevance Model. The Relevance Model increases the weights of these terms, queries for new documents accordingly, recomputes relevance values for the resulting documents via the forward computation, and passes the top n relevant documents on to the Similarity Model.

3. **Manipulating the "Relevance" Slider**: An *Update* message, containing the document ID and relevance parameter, is sent to the Relevance Model that triggers its inverse relevance computation, followed by forward re-execution of the pipeline.

Use Case Using the synthetic intelligence analyst dataset mentioned when StarSPIRE was introduced, we demonstrate semantic interactions on a document collection with Cosmos.

As in StarSPIRE, Cosmos initializes to a blank Visualization, requiring the analyst to search for information. The analyst knows that there is a suspicious person by the name of "Ramazi" and wishes to learn more about him. Searching for "Ramazi" brings 5 document nodes onto the screen, shown in Figure 16-a. By double-clicking on the nodes, the analyst can read their contents in the data panel to the right of the WMDS projection.

After reading the documents, the analyst decides that the document labeled "cia11" contains the most interesting information, describing Ramazi's connection to the Taliban and explaining that he is traveling under an alias. Wanting more information like this, the analyst increases the value of the Relevance Slider for this document, which triggers an inverse computation in the Relevance Model as well as an implicit query. As a result, more documents are brought onto the screen (shown in Figure 16-b).

While the node for "cia11" has moved to the bottom of the screen separated from other nodes, the analyst can begin investigating the new nodes brought onto the Visualization, starting with the nodes closest to "cia11." Though the closest node ("cia2") contains only a vague threat, the next closest node ("cia6") describes how there are missing Stinger missiles that belonged to the Taliban. Since "cia6" contains information like what the analyst is looking for and "cia2" doesn't, the analyst can express interest in distinguishing between these kinds of documents by dragging "cia11" and "cia6" closer together and dragging "cia2" farther away (as depicted in Figure 16-c). After clicking "Update Layout," the inverse computation in the Similarity Model is executed and another implicit query is triggered. The result of this interaction (shown in Figure 16-d brings "cia4" onto the screen and positions it close to "cia11." Investigating this document reveals the name and alias of another suspicious person.

This example demonstrates semantic interactions for sensemaking with text. Although these interactions are simple for the analyst to perform, the complexity in how Cosmos interprets these interactions helps the analyst forage for information through explicit querying (searching for a term like "Ramazi"), implicit querying (caused by dragging observations or increasing the relevance of a document), or discovering potentially useful information through node position and size. Synthesizing information is also supported through the visualization of document nodes, reflecting how documents relate to each other based on analyst interactions.

Elasticsearch

Motivation The two examples presented thus far demonstrate analysis of data sets stored locally. While this can be useful for experimenting with different visual encodings, this does not reflect many real world analysis scenarios. Here, we present a pipeline for analyzing text documents stored in an external database. Such a pipeline enables researchers to investigate methods to bring Big Data into semantic interaction-enabled applications, and will also drive modifications to Models to handle larger datasets. For example, the inverse WMDS algorithm in our Similarity Model has a time complexity of $O(n^2p^2)$ for n observations and p attributes. This time complexity severely limits the number of observations and attributes that can be processed.

To build this new pipeline, we began with the Cosmos pipeline, exchanged the CSV Data Controller for an Elasticsearch Data Controller, and added the Term Frequency Model to the pipeline. The Relevance and Similarity Models required minor modifications to support new assumptions with respect to the data. The details of these changes are described next.

Pipeline The pipeline for this Elasticsearch prototype is shown in Figure 17.

Elasticsearch Data Controller: Enabling Elasticsearch required a new Data Controller to connect to the external search engine rather than loading local data on startup. In order to have complete control over this search engine, we implemented our own basic text search engine using Elasticsearch[36]. After connecting to our Elasticsearch

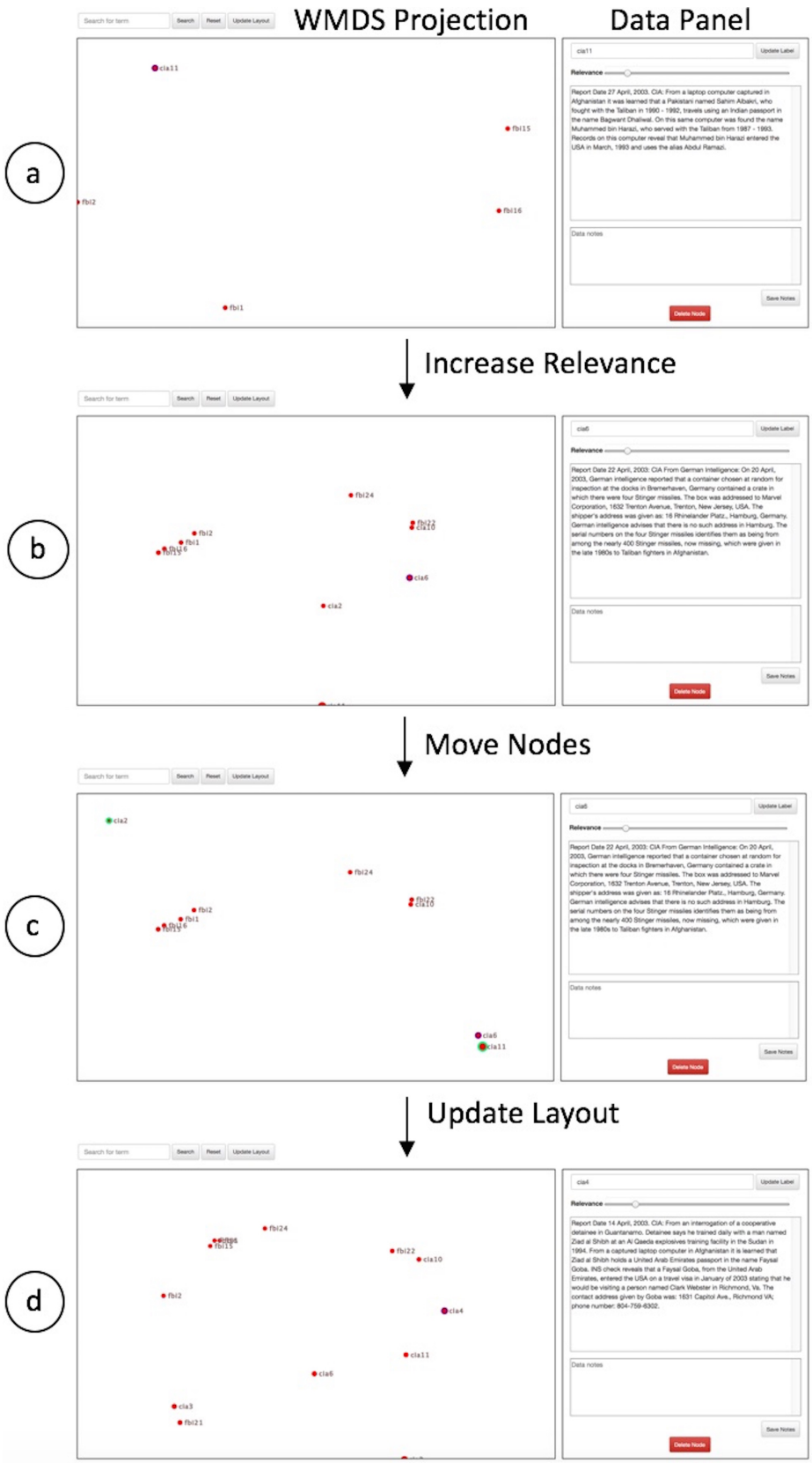

Figure 16. With semantic interaction, analysts can use Cosmos to assist their sensemaking tasks. After an initial search for "Ramazi," an analyst views the matching documents (shown in panel **(a)**). After determining that the document "cia11" is the most relevant document to the investigation, the analyst drags up the "Relevance" slider. This results in an inverse computation in the Relevance Model and an implicit query (results shown in panel **(b)**). After viewing the new documents that were brought into the Visualization, the analyst drags document nodes to differentiate documents like "cia11" and "cia6" from documents like "cia2" (shown in panel **(c)**). After clicking "Update Layout," an inverse computation is triggered in the Similarity Model, and another implicit query is performed (results shown in panel **(d)**).

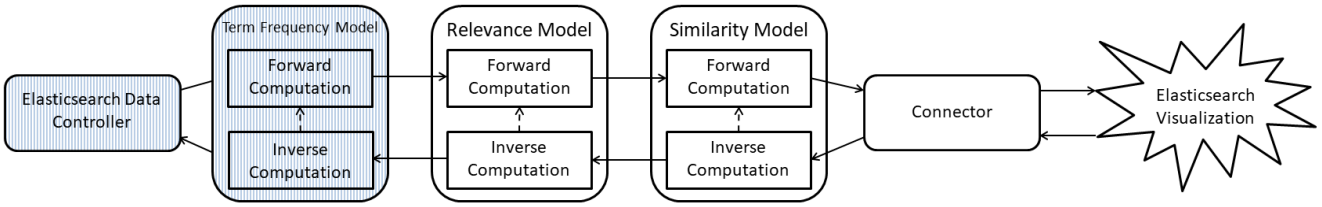

Figure 17. To transform the Cosmos pipeline into Elasticsearch, we add an Elasticsearch Data Controller and a Term Frequency Model to enable the connection to an external search engine and to calculate term frequencies dynamically. The shaded, vertically-striped blocks represent entirely new components.

instance, the Data Controller could retrieve documents in two ways. The first method operates through simple text queries; if a text query interaction arrives at the Data Controller, this query will simply be forwarded to the search engine to retrieve new documents. The second method is through a set of dimension weights sent from the Relevance Model. These dimension weights determine which entities are most relevant, which in turn are used to query the search engine.

Working with an external service eliminates the static data assumptions used by Web Andromeda and Cosmos; the Elasticsearch Data Controller enables adding new documents into the external store while the pipeline is running. Interactions that occur after these documents are added may then pull these new documents into the Visualization. For the purposes of our examples discussed here, we do not highlight this feature. Instead, we focus on the flexibility of the pipeline, which allows us to easily switch the CSV Data Controller with the Elasticsearch Data Controller.

In contrast to the CSV Data Controller, the Elasticsearch Data Controller is only responsible for retrieving new documents either in response to a query or in response to a *Get* message. Data preprocessing is instead handled by the new Term Frequency Model.

Term Frequency Model: Because the Elasticsearch Data Controller only passes raw text data to the pipeline, no preprocessing has occurred on the data. The Term Frequency (TF) Model is required to perform entity extraction and compute entity term frequencies. To compute TF, we use the raw counts for a given term for each document normalized by the maximum number of times that word appears in any of the documents (simple, but sufficient for the purposes of creating a working prototype). These TF values are then used as the attribute values for each document, and are passed down the pipeline to the Relevance Model. Because the Term Frequency Model represents a straightforward data conversion and does not have parameters to learn, it does not have a meaningful inverse computation. Thus, this model uses a simple no-op as its inverse.

Use Case Since Elasticsearch reuses the same Visualization, the same semantic interactions are enabled in this pipeline as well. Thus, with the same synthetic intelligence analyst dataset uploaded into Elasticsearch, an analyst can perform an identical analysis with this Elasticsearch pipeline as with the Cosmos pipeline. The Cosmos Use Case shown in Figure 16 therefore depicts how this Elasticsearch prototype works as well.

Cosmos Radar

Motivation Ruotsalo et al. propose the Intent Radar[22] as a mapping for similarity and relevance in a visualization for data foraging. Within this interface, documents are mapped onto the Radar based on their relevance to analyst searches and to their similarity to each other. More relevant documents will be closer to the center of the Radar, while similar documents will have a similar angle around the Radar. After performing a user study, Ruotsalo et al. found that this new interface enabled analysts to search through the data more quickly and efficiently than interacting with a list of keywords or traditional query searching. To demonstrate the applicability of our pipeline to the field, we re-implement Ruotsalo et al.'s Intent Radar using our pipeline, requiring only a relatively small change to the Cosmos pipeline.

Pipeline Our Cosmos Radar pipeline is shown in Figure 18. We discuss the changes to the Similarity Model and Visualization here.

Similarity Model: Since we use only one dimension to denote similarity instead of two dimensions as in Cosmos, we altered the WMDS Similarity Model to reduce dimensionality to 1D rather than 2D.

Visualization: We modified the Visualization from Cosmos to map similarity to the angle around the Radar and relevance to distance from the center of the Radar, as suggested by Ruotsalo et al. These are trivial changes to how similarity and relevance are visually encoded.

Use Case Except for the dimensionality change in the Similarity Model, Cosmos Radar uses the same pipeline as Cosmos; the biggest change in this pipeline implementation is in the Visualization. To demonstrate how this Cosmos Radar visualization alters the manner in which analysts perform sensemaking tasks, we use the same synthetic intelligence analyst dataset used in Cosmos.

As with Cosmos, Cosmos Radar begins with a blank Visualization, prompting the analyst to search for a term. Searching for "Ramazi" produces the Visualization shown in Figure 19-a. Since the analyst has only performed this one interaction, Cosmos Radar considers all these Ramazi-related documents to have similar relevance calculations, as represented by the distance of each document from the center of the Visualization. Similarity between documents is represented by the angle the document node lies around the Radar. After evaluating the documents, the analyst still decides that "cia11" is the most interesting and increases the Relevance Slider for this document. As a result, the inverse computation is triggered in the Relevance Model along with

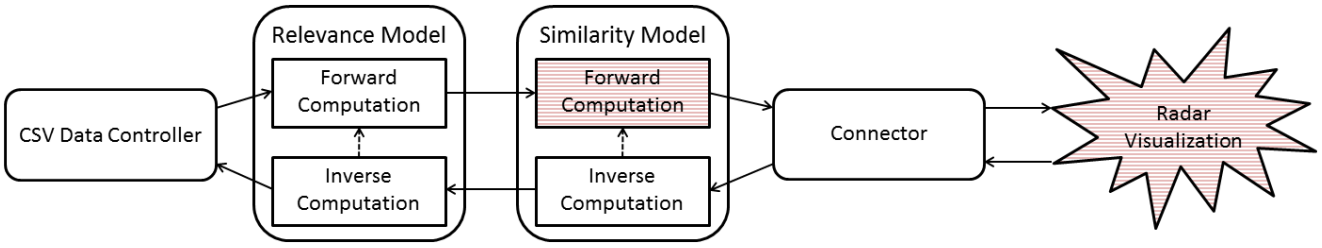

Figure 18. To transform the Cosmos pipeline into Cosmos Radar, the forward computation of the Similarity Model and the Visualization required modification. The shaded, horizontally-striped blocks represent components that previously existed but were modified from the Cosmos implementation.

implicit document querying. The resulting Visualization is shown in Figure 19-b.

This projection of the data does not show any other documents that are highly relevant besides "cia11." The analyst begins to evaluate each document to decide which documents are most relevant to the investigation. Finding that document "cia6" references missing Stinger missiles that belonged to the Taliban, this document can be considered highly important. In contrast, "cia2" does not contain any information that seems immediately important, as it only references a vague threat. To indicate the similarity and relevance of "cia11" and "cia6" as opposed to "cia2," the analyst can drag the two similar nodes together and "cia2" farther away, as depicted in Figure 19-c. After clicking "Update Layout," the Similarity Model performs an inverse computation and an implicit query is performed. The final resulting Visualization is shown in Figure 19-d.

Because the representation of similarity and relevance has changed in this Visualization, we can see how this changes the manner in which the interaction is interpreted. Figure 19-d shows that, while new documents were brought onto the screen, "cia2" has a higher relevance. None of the new documents are plotted very close to "cia11" and "cia6," giving the analyst more options to determine which documents should be inspected next. From this Visualization, the most similar new document is "cia3," which mentions forged passports and likely aliases being used. However, given that the angle between "cia3" and "cia6" is rather large, the analyst may opt to investigate "cia10" instead, if this analysis was not previously done, as "cia10" is much closer. In either case, this is a different conclusion than what was drawn from Cosmos, but neither conclusion is necessarily wrong, as "cia3," "cia10," and "cia4" all hint to other individuals using aliases just like Ramazi in "cia11." The analyst can continue taking an alternative path through the dataset, exploring connections that were not as apparent in Cosmos.

Cosmos Composite

Motivation After observing how the Data Context Map[37] by Cheng and Mueller tags the projection of observations with the attributes, we noted that this feature can help analysts understand *why* observations were plotted in specific locations. The Data Context Map also gives more information at a glance about *how* observations relate to each other than what Web Andromeda and Cosmos provide. We implemented a version of this technique using our pipeline. We refer to our implementation as Cosmos Composite.

By reusing the pipeline components from Cosmos, we are able to extend the Data Context Map methodology with semantic interactions, and also apply it to querying text data. As a result, the components from the Cosmos pipeline must be adjusted so that attributes may be visualized in this Cosmos Composite Visualization. This required adjustments to the Relevance Model, Similarity Model, and the Data Controller. We describe the components of the Cosmos Composite pipeline in detail next.

Pipeline Our pipeline for the Cosmos Composite prototype is depicted in Figure 20.

CSV Data Controller: Since the CSV Data Controller in Cosmos only passes data relative to documents down the pipeline, the Data Controller must be altered to send attribute data as well for the Cosmos Composite pipeline. This allows Models to use this data and, ultimately, for the attributes to be visualized with the observations in the same projection.

Relevance Model: Although we reused much of the same functionality in the Cosmos Relevance Model, some aspects had to be changed to support the Cosmos Composite Visualization. Since observations are now visualized alongside attributes, the queries that the Relevance Model makes to the Data Controller sometimes request attribute data as well as observation data. Like the observations, only the top n attributes are kept. No modification were necessary to the forward and inverse computations of the model; we simply needed to update the message passing to support attribute information. After running the forward computation to determine the relevance values for the observations, the final attribute and observation data are passed to the Similarity Model for additional processing on their way to the Visualization.

Similarity Model: Because we are now visualizing both observations and attributes in the same space, the forward computation of the Similarity Model from Cosmos must be altered. First, the Composite Matrix is constructed following the instructions from Cheng and Mueller[37]. This Composite Matrix represents pairwise distances between observations, between attributes, and between attributes and observations. The dimension weights are used in the creation of this matrix to reflect the importance of the attributes. As a result, the Composite Matrix is a large matrix containing all necessary pairwise distances for MDS. After running the Composite Matrix through this forward computation, the resulting low-dimensional coordinates are passed to the Connector and, ultimately, to the Visualization.

While the Visualization now contains nodes for both observations and attributes, the inverse computation only

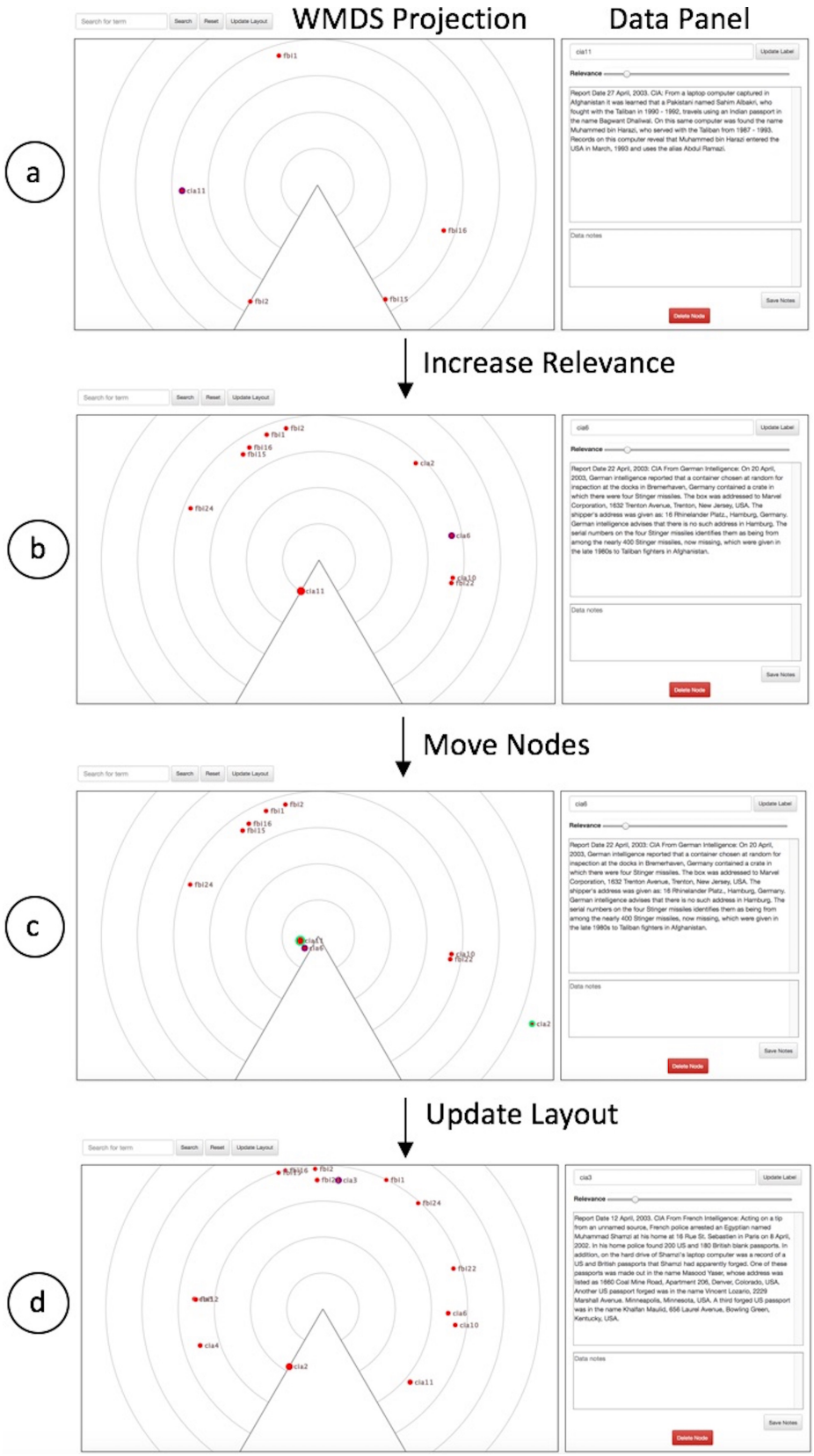

Figure 19. After an initial search for “Ramazi,” the same 5 document nodes appear in the Visualization (shown in panel **(a)**). The nodes are now plotted by mapping relevance to the distance from the center of the Radar and similarity to the angle around the Radar. The analyst increases the “Relevance” slider for “cia11” as before. This results in the same documents being brought onto the screen as in Cosmos, as depicted in panel **(b)**. However, the difference in how these documents are represented in Cosmos Radar gives the analyst an alternative path to determine which documents to inspect next. After inspecting the documents, “cia6” is determined to be similar to “cia11,” while “cia2” does not have the kind of information the analyst is looking for. By dragging the nodes as shown in panel **(c)** and clicking “Update Layout,” new documents are brought onto the screen (shown in panel **(d)**). Although this is a similar set of documents to what was displayed in Cosmos, the difference in the mapping of relevance and similarity to the positions of the nodes continues to enable a different exploration path in the data.

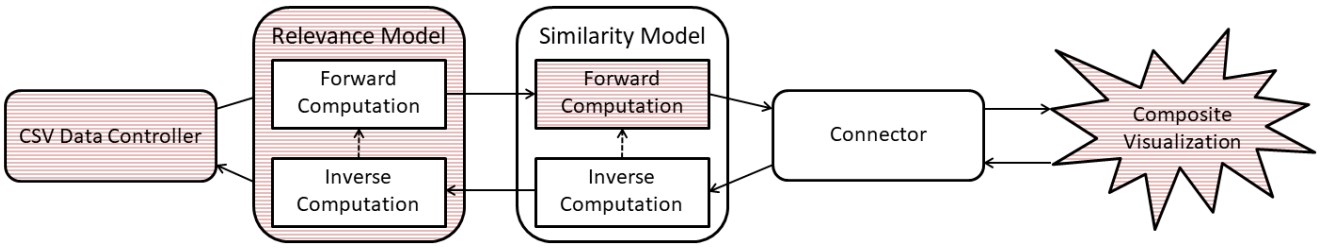

Figure 20. To transform the Cosmos pipeline into Cosmos Composite, the Relevance and Similarity Models, the CSV Data Controller, and the Cosmos Visualization required some modification. The shaded, horizontally-striped blocks represent components that previously existed but were modified from the Cosmos implementation.

handles V2PI with observations. This is because V2PI with attributes would imply a manipulation of a set of weights on the documents. Since we do not have this set of weights, V2PI can only be performed to determine a new set of dimension weights. This means that the Similarity Model ignores V2PI on the attributes. Thus, the inverse computation in the Similarity Model works the same as in Cosmos to compute a new set of dimension weights that display the analyst-defined similarity between the observations. When the forward computation is executed again, these new dimension weights are used to recreate the Composite Matrix.

Visualization: The Cosmos Composite Visualization is nearly identical to the Cosmos Visualization. The only notable differences are that the attribute nodes are a different color than the observation nodes and that the attribute nodes are all the same size since they do not have a relevance calculation associated with them (Figure 21).

Messages: The most notable change in how the messages are interpreted by the pipeline is seen when an analyst drags data points. This interaction is still communicated to the Similarity Model via an *Update* message that indicates which document nodes (not any attribute nodes) have been moved by the analyst. After performing its inverse computation, the Similarity Model then passes the new dimension weights to the Relevance Model. The Relevance Model then performs implicit querying for the analyst. While this brings in new documents, this is also the only case in which new attributes are displayed onto the screen. Therefore, the Document Controller responds to the Relevance Model's query with both document data and attribute data. The Relevance Model ignores the attribute data to determine the relevance calculations for the documents. After determining which documents and which attributes should be passed along to the Similarity Model, the Similarity Model recreates the Composite Matrix. This is then sent into WMDS to calculate the low-dimensional coordinates for the observations and the attributes. All this information is passed through the Controller to the Visualization, which finally plots the observations and attributes in the same panel.

Use Case To highlight how Cosmos Composite differs from Cosmos, we continue to use the same dataset from the Use Case subsection in the Cosmos section. In this new Visualization, the analyst would again begin by searching for "Ramazi." While the same 5 document nodes appear, an additional node for "Abdul Ramazi" appears. This node represents this attribute in the space and is placed closest to the document that has the highest TF-IDF value, "fbi15." This is shown in Figure 21-a. Although the analyst's attention is initially drawn to "fbi15," further investigation of these documents reveals that "cia11" is still the most interesting document to the analyst.

After the analyst increases the relevance for "cia11," an inverse computation in the Relevance Model is triggered along with an implicit query. However, the result of this interaction is that the matching documents and the most relevant attributes are brought onto the screen, as seen in Figure 21-b. This time, "cia6" is plotted closest to "cia11," which draws the analyst's attention directly to this document of interest. After reading this new document and wanting to continue following clues about the Taliban from "cia11," the analyst decides to differentiate "cia11" and "cia6" from "cia2," as this document only contains a vague threat rather than Taliban-specific information (shown in Figure 21-c).

After moving these three nodes and clicking "Update Layout," an inverse computation in the Similarity Model and an implicit query are triggered. The result of this interaction is shown in Figure 21-d, which shows that "fbi22" is also similar to "cia11." Investigation of this document reveals another individual who is also traveling under an alias. Other documents plotted near "cia11," such as "cia4" which was just brought onto the screen, provides new information on other individuals using aliases.

Discussion

Now that we have demonstrated both how our new pipeline can represent existing semantic interaction-enabled visualizations as well as implemented our our pipelines and corresponding suite of visualizations, here we discuss the implications of our new definition of a pipeline.

Rapid Prototyping to Explore Design Trade-Offs

The ability to rapidly prototype several techniques from the visual analytics literature, and augment them with semantic interaction, can enable researchers to explore many design trade-offs. The differences in how the Cosmos Radar Visualization represents document nodes and interprets user interactions compared to Cosmos demonstrates a set of trade-offs. In Cosmos, Similarity was represented through the use of two spatial dimensions and thus seemed to be better than Cosmos Radar at capturing the similarity between documents. However, the fact that Cosmos Radar takes one of those spatial dimensions and uses it to represent relevance means that analysts can immediately tell which documents

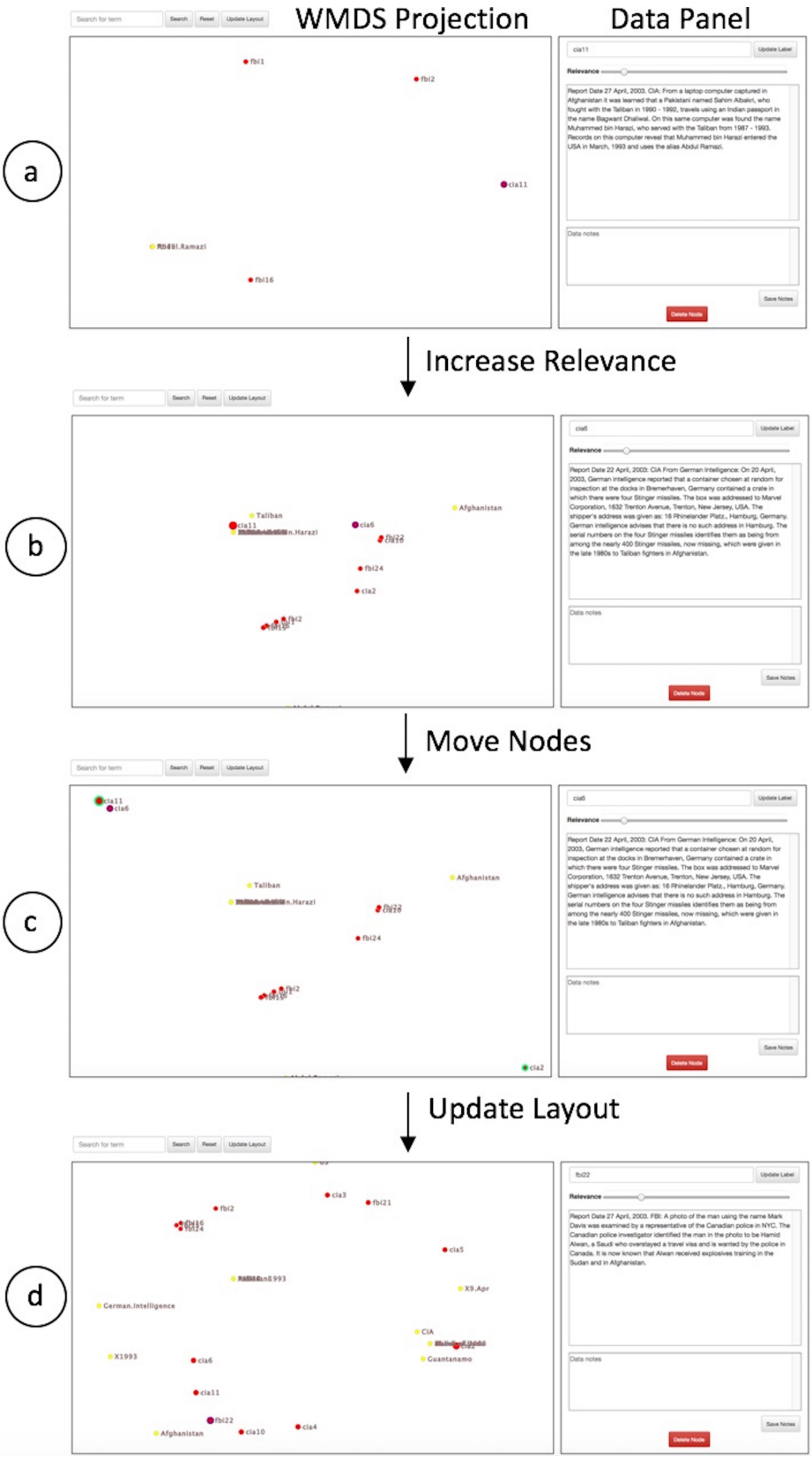

Figure 21. In Cosmos Composite, the analyst can perform the same semantic interactions as in Cosmos. However, the Visualization displays attributes as well as observations, which gives the analyst more feedback on the given interaction. The first example of this feedback is demonstrated by the "Ramazi" attribute being placed close to "fbi15" (actually obscuring the "fbi15" node) after searching for this term, as shown in panel **(a)**. The placement of this attribute can be explained by the fact that "fbi15" has the highest TF value for this entity. Still deciding that "cia11" has the most interesting information, the analyst can then drag up the "Relevance" slider for this document, producing the Visualization in panel **(b)**. After evaluating new documents, the analyst can continue to search for new information like that contained in "cia11" by dragging "cia6" closer to it (as it contains information on another individual traveling under an alias) and dragging "cia2" farther away (depicted in panel **(c)**). After clicking "Update Layout," the final Visualization shown in panel **(d)** helps the analyst find additional documents containing information about individuals with aliases, such as "fbi22" and "cia4."

are most relevant, as this information is now relayed through node position instead of just node size.

Likewise, the Cosmos Composite prototype provides another demonstration of the flexibility of our pipeline implementation. Although the Visualization is nearly the same as that of Cosmos, the updates to the Similarity Model allow the WMDS projection of the observations to be tagged with the attributes. This provides context to the Visualization, which helps the analyst determine where observations of interest may lie. Thus, this tagging of the space assists the analyst with their foraging and sensemaking tasks. This Visualization, combined with the semantic interactions, provides new and intuitive methods for performing sensemaking tasks.

Introducing an Elasticsearch Data Controller (and the related Term Frequency Model) further demonstrates the flexibility of this modular pipeline implementation. The Use Cases between the Cosmos pipeline and Elasticsearch pipelines were identical, with the changes to the backend and the storage location of the documents entirely hidden from analysts interacting with the Visualizations.

Pipeline Evaluation

Our stricter definition of semantic interaction based on the idea of a sequence of composable models, each containing forward and inverse computations, that are arranged into a bidirectional pipeline served our prototype examples well. Each of the semantic interactions included in the tools was enabled by an interaction afforded by the Visualization and handled by one or more Models (and occasionally the Data Controller). The seven principles for semantic interaction defined by Endert et al.[3] are still supported by these interactions as well, further justifying the refined definition of semantic interaction that we introduce in this work. Our definition better operationalizes semantic interaction in hopes of enabling more research and development.

The commonalities that we identified in existing semantic interaction-enabled applications also justify our design. We were able to replicate Andromeda[15] in our Web Andromeda tool, we used semantic interactions from StarSPIRE[7] to manipulate observations in Cosmos and its Radar and Elasticsearch extensions, and we created an alternative visual encoding Cosmos inspired by Intent Radar[22]. Though we do not do so in this paper, we assert that the additional applications described earlier[19–21;23] can also be modeled and implemented using our pipeline. This is due to the underlying V2PI observation manipulation in each of these applications, which is incorporated in the Similarity Model described in *Web Andromeda*.

Our pipeline met our three requirements to support semantic interaction, as well as our four additional requirements for a web-based, rapid prototyping implementation. The implementation of our Models and model communication guidelines allow for chaining a sequence of Models together. To support model inversion, each Model must contain an inverse computation to match its forward computation, even if the inverse computation simply passes data directly through. Chaining the forward models together from Data to Visualization and the inverse models together from Visualization back to Data supports bidirectionality.

The additional rapid prototyping requirements are also seen in our implementation, as well as justified by our prototypes. By implementing each Model as a standalone class, we were able to freely exchange components and add new components to concrete pipelines. These are shown by our ability to convert Web Andromeda into Cosmos through the addition of a Relevance Model, and then into Elasticsearch by replacing the Data Controller. The ease of swapping components to create new Visualizations and new prototypes is described at the Model level rather than at the individual lines of code level, as we designed the Models to be self-contained. Many of the modifications made to the Models in our prototypes did not amount to more than a few lines of code. The Connector separates the Visualization from the remainder of the pipeline, separating the user interactions from the details of the Models. Creating D3 visualizations accessible from the Web met our final goal.

With our pipeline, researchers are able to define their own Models and insert them into pipelines of new prototypes, thereby addressing further challenges and research questions in semantic interaction. Some proposed questions that this pipeline could help address are described next.

Answering Research Questions on Visual Analytics with Semantic Interactions

By continuing to build upon our current prototypes or creating new ones, we can answer additional questions surrounding semantic interaction in visual analytics tools. Some future research questions that we pose are:

How can we combine multiple types of data into a single semantic interaction-enabled pipeline and, eventually, into a single Visualization? In our visual analytics prototypes, numerical data (or text data that has been converted into numerical data) is the only type of data used. However, focusing on only one type of data in a visual analytics tool severely limits the analyst's ability to perform sensemaking tasks that combine different types of data. For example, an analyst may wish to track a particular person in an airport through security videos of an airport and logs for flights leaving the airport. Although the analyst can search for the person's name in the flight logs, it is harder to connect this information to the security videos.

Furthermore, how would the security videos and flight logs appear in a Visualization? In our visual analytics prototypes, we assume that every observation can be compared to any other observation through a similarity metric (i.e., distance function used for WMDS in the Similarity Model). Therefore, in order to extend our visual analytics prototypes to handle multiple types of data, we must update the Data Controller as well as update our Similarity Model to compute the similarity between any two observations (i.e., between two flight logs, between two security videos, and between flight logs and security videos).

How can a bidirectional pipeline support streaming data? In order to support streaming data on any level, we must first create a Data Controller that is capable of handling such data. This Data Controller must be able to retrieve data as it becomes available from the data source and push relevant data through the pipeline without being triggered by a user interaction. This push behavior is a new mechanism

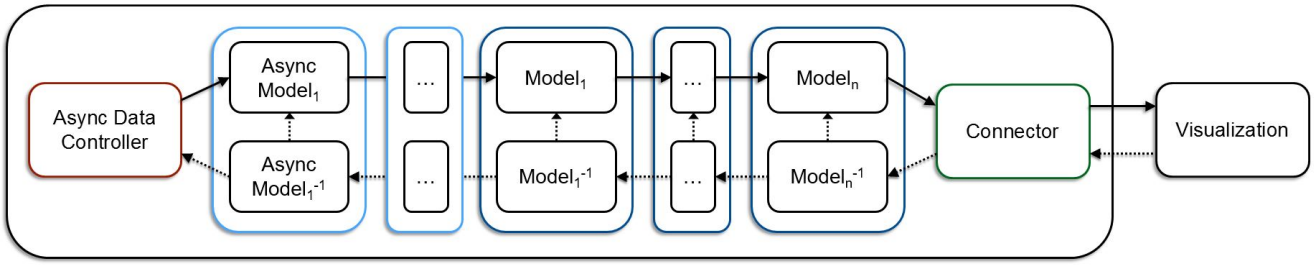

Figure 22. A pipeline with asynchronous Models, for potential use in applications that deal with streaming data.

that must be added to the pipeline; retrieving streaming data from a data source can be handled in a manner similar to the Elasticsearch Data Controller that checks for updates at regularly-timed intervals. The Connector must also support pushing data to the Visualization. Certain approaches to implementing a Connector may make this impossible, such as those based on Remote Procedure Call (RPC) protocols[38] that can only respond to outside requests. Therefore, a new Connector is necessary for a streaming data pipeline.

Additionally, pushing data involves running the pipeline over multiple threads. In order to provide simple background processing capability, an asynchronous type of Model could be created. The structure of a pipeline using asynchronous models can be seen in Figure 22. A simple background threading system results from grouping a series of asynchronous Models followed by a series of synchronous Models. Upon an interaction, the synchronous Models are executed as previously described. When an asynchronous Model is reached along the inverse path, the input data containing the interaction and any new Model parameters are added to a queue. The forward computations, starting with the first synchronous Model, are then immediately run, and the Visualization is notified of any updates. In a separate thread, the asynchronous Models are run in a similar manner to the original pipeline. Each inverse computation is executed in series until the Data Controller is reached or a Model short circuits. The forward computations are then run until the first synchronous Model is reached. The forward computation of this first Model is run using the data passed down through the asynchronous Models, and the background thread then takes the next item off the queue and iterates again.

The main drawback of this approach is the updates from each interaction may not contain the most up to date data. The synchronous part of the pipeline must return a response back before the asynchronous portion completes, meaning no new data resulting from the asynchronous Models will be returned in the initial update. Algorithms of an asynchronous model could be run multiple times concurrently due to multiple pushes, or from a push and the asynchronous background thread described previously. An alternative approach to asynchronous Models involves parallelization, allowing multiple Models to execute their computations at the same time on independent processor cores rather than using the serial behavior shown in all of our example applications. We save this for future development.

How can the models of visual analytics tools process real-time data in a bidirectional pipeline while allowing for user interaction? The Models in our visual analytics prototypes assume that any information passed to the Visualization is either the result of initializing the Pipeline and Visualization or from an interaction. That is, with real-time data, we currently have no method of being able to push data to the Visualization without an interaction to trigger this push. Thus, we must alter our Models to enable them to push data up to the Visualization as new data enters the pipeline.

A push can originate from either the Data Controller or from a Model as a function call. Pushed data is sent through all or a portion of the forward pipeline, just as data would progress during a user interaction-triggered update. Models need not know whether they are processing push-generated data or interaction-generated data. Once the forward pipeline is complete, the results are pushed to the Visualization.

How can streaming data be visualized without causing a loss of context? After enhancing our Models to enable pushing data to the Visualization, we must consider how to update the Visualization without causing a loss of context. That is, if data is going to be pushed to the Visualization automatically rather than in response to the analyst's interactions, we must be careful not to alter the Visualization so much that the analyst cannot keep track of where existing datapoints were located before a push. For example, in our current implementation of Cosmos, the exact position of every node in the Visualization is based on the other nodes. Therefore, adding or removing nodes from the Visualization alters the entire projection. Maintaining an analyst's mental map of a visualization during layout adjustments is a well-studied problem[39].

How can an analyst interact with streaming data that updates the visualization? Building on the previous question, if the Visualization is automatically updating, then not only is it difficult to visually track where nodes are moving, but also it is difficult to interact with these nodes. The interaction mechanism that solves this challenge is directly dependent upon the solution found to the previous question.

How can 2D visual analytic prototypes be extended into 3D or immersive environments? While the richness of 2D web visualization is growing, so too are the capabilities for interactive 3D visualizations. Extending a 2D visual analytics prototype into 3D requires alterations to both the Similarity Model and the Visualization. Although the Similarity Model for the 3D Visualization is conceptually the same as the Similarity Model used in 2D Cosmos, the low-dimensional coordinates returned by the 3D Similarity Model must return 3D coordinates instead of 2D coordinates in order to map the document nodes into 3D space.

Following this alteration to the Similarity Model, we only need to create a new 3D Visualization that can communicate with the rest of the pipeline. Figure 23 shows an initial

Figure 23. A screenshot of the 3D web visualization. The layout of the web page is based on the layout in the original 2D visualization. With minor changes to the base implementation of the pipeline, we are able to graph the data points represented as spheres in 3D.

prototype of a 3D web-based visualization built in X3DOM. In this new Visualization, the data points are represented as spheres instead of circles, using the radius of the sphere to encode the relevance of each document.

What semantic interactions are enabled by 3D or immersive environments? We can also begin exploring how we can interact with the visualizations in 3D. Our focus in particular lies in exploring the semantic interactions afforded by 3D environments. How does an extra dimension of space affect how analysts interpret and interact with the Visualization? How do we adapt interactions in 2D Visualizations to 3D? Are there new interactions that are enabled through en extra dimension of space?

In the 3D prototype from Figure 23, we began to experiment with a variety of interaction options. Instead of double-clicking a node to populate the data fields to the right of the graph, we chose to use right-clicking. To handle new interactions for exploring the 3D space, there are also buttons to view the scene in multiple angles. These different angles include the Top, Bottom, Front, Back, Left, and Right views. The "Reset View" button resets the view to the default viewpoint. Apart from these, all other interactions are pulled directly from the original 2D prototype.

Limitations

Despite the power and flexibility of our proposed semantic interaction pipeline requirements and concrete implementation, neither is without limitations. We briefly address several of these limitations here.

Requirements Limitations The primary limitation of our proposed new pipeline that enables semantic interaction lies in the requirement of providing an inverse computation for each forward computation. We assert this requirement as essential for enabling semantic interaction, yet we provide no guidance for how to determine what such an inverse computation should be. That is, the inverse computation can be mathematically rigorous, heuristic, probabilistic, or even an identity function. As noted previously, the Term Frequency Model in our Elasticsearch pipeline implementation employs an identity function as an inverse computation since any other inverse computation does not make sense in our context.

Similarly, our proposed generalized visual analytics pipeline requires Models to be composable, but we provide limited-to-no instructions for defining the communication between models. This is in part a result of the flexibility requirement of the pipeline; we do not want to provide a rigorous structure such that exploration into unusual Model combinations and orderings cannot be explored. Indeed, these explorations motivate potential future work directions, such as the most computationally efficient or semantically-appropriate ordering of Models and interactions.

Implementation Limitations One limitation of our concrete pipeline implementation of this pipeline results from our rapid prototyping design decision. Rather than creating fully-featured tools, we use this pipeline to quickly and efficiently prototype visual analytics tools to explore the semantic interaction design space. As a result of this design decision, many of the prototypes that we implemented in the previous section appear visually similar and only support a limited number of semantic interactions. However, we argue that each of our prototypes can support additional semantic interactions with the addition of more models to each pipeline. Furthermore, the visual similarity of our pipelines was intentionally designed to highlight the internal flexibility of our prototyping system, changing the input data and interactions with only limited changes to the code.

Conclusion

In this work, we began by examining a number of visual analytics applications that make use of V2PI. Through this examination, and influenced by the Sensemaking Process described by Pirolli and Card[9], we proposed three characteristics shared by semantic interaction applications: a series of models, looping foraging and sensemaking interactions, and inverse interactions. These characteristics were then formulated as three matching concrete structures in a proposed pipeline for semantic interaction: model composition, bidirectionality, and model inversion. From these requirements, we proposed a new visual analytics pipeline that enables proper representation of the complexity involved in semantic interactions such as V2PI.

We developed a modular, bidirectional pipeline for creating visual analytics prototypes that use semantic interaction and V2PI to aid the Sensemaking Process. Four key pieces make up this pipeline. A Data Controller defines what type of data is being visualized and how it is accessed. A series of Models transform the data into a form suitable to be visualized, as well as interpret interactions from the visualization. Finally, a Connector controls how the Visualization communicates with the rest of the pipeline. Each of these pieces can easily be replaced to quickly prototype and experiment with different types of data, mathematical models, semantic interaction techniques, and visual encodings.

We demonstrated the flexibility of our pipeline implementation by developing several prototypes that exemplify how to research each of these aspects. We discussed the motivation behind each of the prototypes, described the new and reused components of each pipeline, and provided a use case for each pipeline.

By enabling rapid prototyping of such tools, researchers will be able to quickly conduct user studies on many of these alternative methods of semantic interaction. We intend to continue expanding on these prototypes and conduct our own user studies. These studies will reveal how the analyst

perceives these different visual encodings and interactions, which methods best support the analyst's sensemaking process, and how to develop better visual analytics tools in the future. We hope that our new visual analytics pipeline will refine the traditional pipeline to emphasize its bidirectional nature and the role of inverse algorithms to interpret semantic interactions.

Acknowledgments

Acknowledgments removed for blind review

Funding

This research was partially funded by General Dynamics Missions Systems; and by the National Science Foundation [grant IIS-1447416].

Declaration of Conflicting Interests

The authors declare that there is no conflict of interest.

References

1. Endert A. Semantic interaction for visual analytics: Toward coupling cognition and computation. *Computer Graphics and Applications, IEEE* 2014; 34(4): 8–15. DOI:10.1109/MCG.2014.73.
2. Endert A, Han C, Maiti D et al. Observation-level interaction with statistical models for visual analytics. In *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*. pp. 121–130. DOI:10.1109/VAST.2011.6102449.
3. Endert A, Fiaux P and North C. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '12, New York, NY, USA: ACM. ISBN 978-1-4503-1015-4, pp. 473–482. DOI:10.1145/2207676.2207741. URL http://doi.acm.org/10.1145/2207676.2207741.
4. House L, Leman S and Han C. Bayesian visual analytics: Bava. *Statistical Analysis and Data Mining* 2015; 8(1): 1–13. DOI:10.1002/sam.11253. URL http://dx.doi.org/10.1002/sam.11253.
5. Leman SC, House L, Maiti D et al. Visual to parametric interaction (v2pi). *PLoS ONE* 2013; 8(3): 1–12. DOI:10.1371/journal.pone.0050474. URL http://dx.doi.org/10.1371%2Fjournal.pone.0050474.
6. Self JZ, Hu X, House L et al. Designing usable interactive visual analytics tools for dimension reduction. In *CHI 2016 Workshop on Human-Centered Machine Learning (HCML)*. p. 7.
7. Bradel L, North C, House L et al. Multi-model semantic interaction for text analytics. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*. pp. 163–172. DOI:10.1109/VAST.2014.7042492.
8. Endert A, Fiaux P and North C. Semantic interaction for sensemaking: Inferring analytical reasoning for model steering. *IEEE Transactions on Visualization and Computer Graphics* 2012; 18(12): 2879–2888. DOI:10.1109/TVCG.2012.260.
9. Pirolli P and Card S. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. *Proceedings of International Conference on Intelligence Analysis* 2005; 5: 2–4.
10. Bradel L, Wycoff N, House L et al. Big text visual analytics in sensemaking. In *2015 Big Data Visual Analytics (BDVA)*. pp. 1–8. DOI:10.1109/BDVA.2015.7314287.
11. Wang XM, Zhang TY, Ma YX et al. A survey of visual analytic pipelines. *Journal of Computer Science and Technology* 2016; 31(4): 787–804. DOI:10.1007/s11390-016-1663-1. URL http://dx.doi.org/10.1007/s11390-016-1663-1.
12. Keim D, Andrienko G, Fekete JD et al. *Visual Analytics: Definition, Process, and Challenges*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-70956-5, 2008. pp. 154–175. DOI:10.1007/978-3-540-70956-5_7. URL http://dx.doi.org/10.1007/978-3-540-70956-5_7.
13. Pirolli P and Card S. Information foraging. *Psychological review* 1999; 106(4): 643.
14. Russell DM, Stefik MJ, Pirolli P et al. The cost structure of sensemaking. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*. CHI '93, New York, NY, USA: ACM. ISBN 0-89791-575-5, pp. 269–276. DOI:10.1145/169059.169209. URL http://doi.acm.org/10.1145/169059.169209.
15. Self JZ, Vinayagam RK, Fry JT et al. Bridging the gap between user intention and model parameters for human-in-the-loop data analytics. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. HILDA '16, New York, NY, USA: ACM. ISBN 978-1-4503-4207-0, pp. 3:1–3:6. DOI:10.1145/2939502.2939505. URL http://doi.acm.org/10.1145/2939502.2939505.
16. Torgerson WS. *Theory and methods of scaling*. Oxford, England: Wiley, 1958.
17. Kruskal JB and Wish M. Multidimensional scaling. *Quantitative Applications in the social Sciences Series, Newbury Park: Sage Publications* 1978; 11.
18. Sacha D, Zhang L, Sedlmair M et al. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Transactions on Visualization and Computer Graphics* 2017; 23(1): 241–250. DOI:10.1109/TVCG.2016.2598495.
19. Brown ET, Liu J, Brodley CE et al. Dis-function: Learning distance functions interactively. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*. pp. 83–92. DOI:10.1109/VAST.2012.6400486.
20. Paulovich F, Eler D, Poco J et al. Piecewise laplacian-based projection for interactive data exploration and organization. *Computer Graphics Forum* 2011; 30(3): 1091–1100. DOI:10.1111/j.1467-8659.2011.01958.x. URL http://dx.doi.org/10.1111/j.1467-8659.2011.01958.x.
21. Mamani GMH, Fatore FM, Nonato LG et al. User-driven feature space transformation. *Computer Graphics Forum* 2013; 32(3pt3): 291–299. DOI:10.1111/cgf.12116. URL http://dx.doi.org/10.1111/cgf.12116.
22. Ruotsalo T, Peltonen J, Eugster M et al. Directing exploratory search with interactive intent modeling. In *Proceedings of the 22nd ACM international conference on Conference on information and knowledge management*. CIKM '13, New York, NY, USA: ACM. ISBN 978-1-4503-2263-8, pp. 1759–1764. DOI:10.1145/2505515.2505644. URL http://doi.acm.org/10.1145/2505515.2505644.
23. Molchanov V and Linsen L. Interactive Design of Multidimensional Data Projection Layout. In Elmqvist N, Hlawitschka M and Kennedy J (eds.) *EuroVis - Short Papers*.

The Eurographics Association. ISBN 978-3-905674-69-9. DOI:10.2312/eurovisshort.20141152.
24. Node.js. https://nodejs.org, 2016. Accessed: 2015-10-24.
25. Socket.io. https://socket.io/, 2017. Accessed: 2017-05-16.
26. zerorpc. http://zerorpc.io, 2016. Accessed: 2016-02-09.
27. Wenskovitch J and North C. Observation-level interaction with clustering and dimension reduction algorithms. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*. HILDA'17, New York, NY, USA: ACM. ISBN 978-1-4503-5029-7, pp. 14:1–14:6. DOI:10.1145/3077257.3077259. URL http://doi.acm.org/10.1145/3077257.3077259.
28. D3. https://d3js.org/, 2016. Accessed: 2015-11-05.
29. Behr J, Eschler P, Jung Y et al. X3dom: A dom-based html5/x3d integration model. In *Proceedings of the 14th International Conference on 3D Web Technology*. Web3D '09, New York, NY, USA: ACM. ISBN 978-1-60558-432-4, pp. 127–135. DOI:10.1145/1559764.1559784. URL http://doi.acm.org/10.1145/1559764.1559784.
30. Websockets. https://www.w3.org/TR/websockets/, 2016. Accessed: 2015-11-05.
31. Hendler J. Web 3.0 emerging. *Computer* 2009; 42(1): 111–113. DOI:10.1109/MC.2009.30.
32. Taivalsaari A and Mikkonen T. The web as an application platform: The saga continues. In *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*. pp. 170–174. DOI:10.1109/SEAA.2011.35.
33. Polys NF, Knapp B, Bock M et al. Fusality: An open framework for cross-platform mirror world installations. In *Proceedings of the 20th International Conference on 3D Web Technology*. Web3D '15, New York, NY, USA: ACM. ISBN 978-1-4503-3647-5, pp. 171–179. DOI:10.1145/2775292.2775317. URL http://doi.acm.org/10.1145/2775292.2775317.
34. Zeromq. http://zeromq.org, 2016. Accessed: 2016-02-09.
35. Lampert CH, Nickisch H, Harmeling S et al. Animals with attributes: A dataset for attribute based classification, 2009.
36. Gormley C and Tong Z. *Elasticsearch: The Definitive Guide*. 1st ed. O'Reilly Media, Inc., 2015. ISBN 1449358543, 9781449358549.
37. Cheng S and Mueller K. The data context map: Fusing data and attributes into a unified display. *IEEE Transactions on Visualization and Computer Graphics* 2016; 22(1): 121–130. DOI:10.1109/TVCG.2015.2467552.
38. Thurlow R. Rpc: Remote procedure call protocol specification version 2, 2009.
39. Misue K, Eades P, Lai W et al. Layout adjustment and the mental map. *Journal of Visual Languages & Computing* 1995; 6(2): 183 – 210. DOI:http://dx.doi.org/10.1006/jvlc.1995.1010. URL http://www.sciencedirect.com/science/article/pii/S1045926X85710105.