Visualizing the Commonalities Between Hierarchically Structured Data Queries

Chad Jones
University of California, Davis
Email: cejjones@ucdavis.edu

Ryan Armstrong
University of California, Davis
Email: rnarmstrong@ucdavis.edu

Kwan-Liu Ma
University of California, Davis
Email: ma@cs.ucdavis.edu

Abstract—**In this paper we present an information visualization approach for visualizing the intersection between multiple feature hierarchies. In cases where a dataset has multiple complex features of interest, especially ones that have been hierarchically clustered, it is often very difficult to grasp the commonalities between them. Understanding the overlap between features can help researchers discover correlations or hot spots in their data. Our method steps back from actual data features, simplifying their representation, and presenting them in a fashion which facilitates interpretation of their inter-feature relationships. We call this new representation a *knowledge graph*, and we show how it can be used to investigate the overlap of multiple features in a photo database and a social network dataset.**

I. Introduction

As data continues to increase in both size and complexity, extracting a single feature from an entire collection of information becomes a useful path on the road toward discovery. Smaller subsets of data are easier to understand and process, using methods such as numerical analysis, hierarchical clustering or direct visualization. Instances arise, however, where several different features are required to adequately study a problem. Besides their individual importance, the separate features of information could potentially contain correlating values that help support or deny a hypothesis. Exploring and understanding feature correlations is a task that can benefit greatly from interactive visualizations.

What a feature represents depends on the underlying data and how the feature is extracted. For social networks, a data query might return communities of friends with a common interest, but for an image database, a feature could be a collection of photos sharing a specific tag. Individually, each query may result in thousands of data items being returned, so when the filtered data becomes overwhelmingly complex, a common approach is to hierarchically cluster results into separate categories that show more detail as one expands each level. Once all the results are collected and processed, a researcher may then want to see how the various resulting feature hierarchies are related. In order to assist in the exploration process and provide a meaningful basis for feature comparison, we propose a representation that can be applied to all types of problem sets

Our approach creates a high-level abstraction of feature information in the form of a *knowledge graph* visualization. In a knowledge graph, hierarchically structured features and their intersections are shown in a multi-level compound hypergraph, allowing researchers to focus on higher level connections between features and providing interactive ways for hiding and revealing details. Unlike recent work in the area of a set visualization, we focus on visualizing sets that are clustered into one or more levels of abstraction using interactive techniques provided by node-link diagrams. The classification of features within a feature hierarchy provides new ways of understanding how different features correlate on multiple levels.

II. Background

The term knowledge graph is chosen for this feature-space visualization approach because the final graph represents the intersection of the user's results that were generated during the exploration process. The knowledge graph is a collection of what the user has discovered and how those features overlap. Our approach is built from areas of visualization meant to show relationships and correlations between result sets.

A. Visualization of Set Relations

The visualization of overlapping features can be traced all the way back to the 19th century when John Venn invented his set diagrams. Euler and Venn diagrams are the most common ways of visualizing the overlap of multiple sets, and they effectively show regions representing inclusion, exclusion, and containment relationships. The simplest representation utilizes overlapping circles. However, this approach is limited by the fact that it cannot accurately show intersections of four or more sets. To solve this problem, there has been considerable research to expand Venn diagrams to handle more possible set intersections [7], [17]. Recently, many algorithms have been developed to generate such diagrams automatically [16], [4].

Verroust and Viaud provided a new constructive method for creating extended Euler diagrams, which relax and extend the strict conditions on the traditional form of the diagrams [21]. With this new method, up to 8 sets can be visualized simultaneously. Using a planar intersection graph as the underlying layout, extended Euler diagrams do not require that regions be convex or completely filled, which allows for more control over the drawing procedure. A visualization system called VennMaster was developed by Kestler et al. to show multiple overlapping sets of results from genetic databases [12]. The authors provide a genetic algorithm to generate and position area-proportional sets and their overlap. The inconsistencies are listed in a separate table that is linked to the visualization.

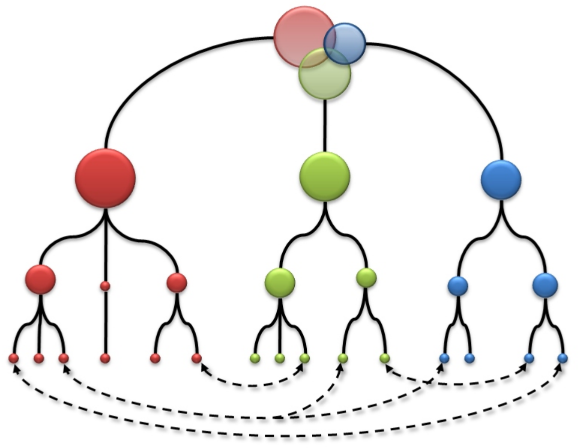

Fig. 1. Each set of data items is hierarchically structured. On every level of the hierarchy, there are adjacency (hyper)edges, shown here with dotted lines, connecting nodes containing items in common. For simplification purposes, only the lowest level adjacency edges are shown in this example.

Since not all set relationships must be visualized using standard Euler diagrams, other techniques have been introduced to extend or complement traditional methods. One such approach, developed by Simonetto and Auber, relies on the underlying intersection graphs for drawing Euler-like diagrams as flexible wrappers [18]. In an intersection graph, all nodes represent regions to be drawn and edges represent zones that are adjacent. The boundary of the sets are then drawn with the intersection graph as a guide, creating blob like structures that overlap where appropriate. The InfoCrystal layout, created by Anselm Spoerri, uses a specialized arrangement of shaped nodes to show every possible set relationship at once [19]. Every data item is assigned to the node that represents the region it belongs to, such as the intersection of sets A and B. Chiara and Fish developed a system called EulerView to replace certain tasks that usually rely on tree views to control and navigate hierarchies [2]. It utilizes a look and feel similar to tree views so users can build and manage their own set relationships. Other systems, such as Mirage [8], leave the data correlation task to the user by providing a suite of various information visualization tools.

Our graph representation shares commonalities with different portions of the previous work in the field. However, we focus on an edge-standard hypergraph representation, which is rarely used for directly visualizing set relations. Unlike a standard binary graph, hypergraphs allow edges that can connect any number of nodes together using a hyperedge. Johnson and Pollack were the first to utilize the commonality between Venn diagrams and hypergraphs, and they showed that determining the planarity of a hypergraph is NP-complete [11]. Harel used a combination of Venn diagrams and hypergraphs to create higraphs, which have been widely used for diagramming concurrent states [6].

Hypergraphs offer many advantages when dealing with the overlap of hierarchically structured sets due to their flexibility and interactivity. Ambiguities resulting from misleading overlap are not a problem in a hypergraph as intersections are discreet edges, and the overwhelming amount of information can be handled in an interactive focus+context manner.

B. Graph-Based Relational Visualization

Graphs are classified into a wide-array of types depending on the kind of relational information being conveyed [14]. The most relevant type of graph for our research is the compound graph, which combines an intersection graph for visualizing graph hierarchy and an undirected or directed graph for showing adjacency relations. Visualizations that show adjacency relations on top of hierarchies include TreeMaps, ArcTrees [13], polyarchies [15], compound digraphs [20], and balloon graphs. Holten and van Wijk improved on these visualizations by introducing hierarchical edge bundling [9]. These represent the common basis for creating visualizations that can represent intersections between two or more distinct hierarchical graphs.

To solve the problem of visualizing node correlation between independent graphs, VisLink expands on the idea of coordinated multiple views by connecting the views with adjacency edges, essentially creating heterogeneous compound graphs [1]. While the visualization only deals with binary connections, it does an efficient job of revealing connections between pairs of visualization results. Along a similar line, Holten and van Wijk created a visualization for showing the relations between a pair of distinct hierarchies [10]. Each hierarchy is laid out separately, and edges are used to connect hierarchies that are related, similar to the way VisLinks connects related nodes between views.

The flexibility and interactivity of these visualizations is what led us to explore the possibility of representing complex set correlations using hypergraphs. We utilize many of the principles found in the compound graph visualizations discussed here and make our own adjustments and additions to meet the needs of the problem.

III. Knowledge Graphs

Our system, KnowHow, utilizes a *knowledge graph* to represent the feature hierarchies generated from a collection of data. In the following sections we will describe the basic concepts that KnowHow is built from and then cover the visual representation and our implementation.

A. Feature Hierarchy

One important part of data analysis and exploration is separating meaningful subsets of data from the entire collection. This process can result in numerous features, depending on the type of data and the extraction method. We define a set to be a collection of data that results from some type of search query, such as an SQL select statement. Unlike previous set visualizations, we focus on sets that are hierarchically categorized.

We define a feature to be a collection of data items, and a feature hierarchy is defined as a tree where features are divided into smaller and smaller groups. The root level is a single feature that combines all of the data from all search queries,

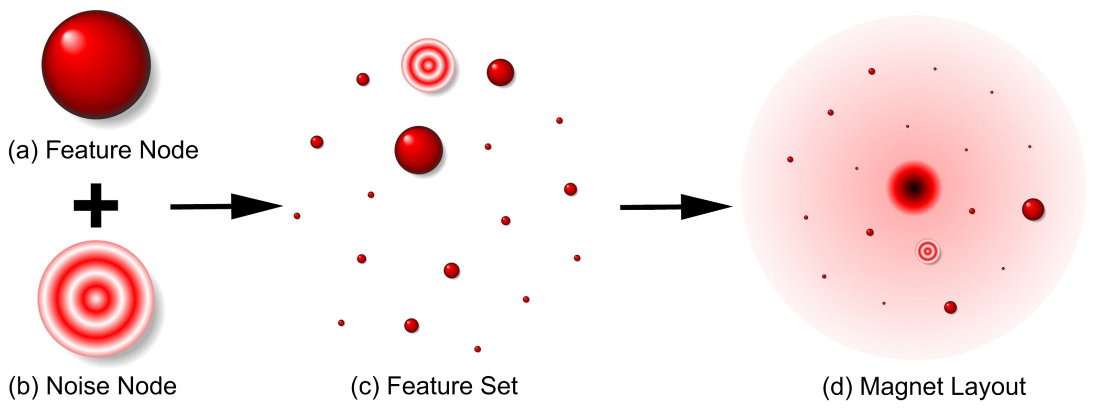

Fig. 2. The nodes of the knowledge graph represent features are colored according to the parent set. A regular feature node is shown in (a) and a specialized noise node is shown in (b). The noise node aggregates small features together. Together, nodes from the same level in the hierarchy form a feature set (c). The nodes of the feature set are positioned with a force directed layout and are influenced by its movable feature magnet, which is the center icon of (d).

the first level represents the original high level features, where each query has a single node. The leaves represent degenerate features, where only a single data item is represented by that node. The middle levels contain features generated by hierarchically categorizing the resulting data. We refer to each collection of features on a given hierarchy level as a feature set. Figure 1 shows three query sets that each have one hierarchical level of division.

The basis of our knowledge graph representation, however, is to show the overlap between features from multiple search queries. Since degenerate features, or the leaves, may be found in more than one search query, we connect duplicate leaves with an adjacency edge or *hyperedge*, which are shown as dotted lines at the bottom of Figure 1. These adjacency edges are propagated up the tree so that each feature in the tree is connected to other features with shared items. The structure is similar to the one visualized by Holten with hierarchical edge bundles [9], with the addition of hyperedges.

The grouping of data items into feature hierarchies depends on the what the user deems important, and each type of data can have different attributes used for dividing up the results. For a social network, a set would contain a collection of individuals who fall into a certain search query, such as users from the same university. Calculating community structure is a common method for dividing the results into a feature set. In such an example, an overlap of features exists if two community clusters contain the same person. Other types of simple categorization could hierarchically divide data by geographical location or by date and time.

B. Hypergraph Representation

Given that each feature hierarchy constitutes a collection of clustered results, the relationships between those results at a given tree level can be represented by a hypergraph. In a knowledge graph, each node represents a clustered feature and the edges convey the intersections that exists between the features.

1) Nodes: At the first level of a feature hierarchy, each result subset is represented by a single node. We assign an individual color to each of these nodes, such that all features originating from it will share that color. This high level node is split into multiple feature nodes as the user descends the tree, revealing new feature sets at each level. In addition to coloring, we collect a family of features from the same result subset into a cohesive magnet cloud. The size of a node is proportional to number of data items contained within that clustered feature.

We also allow for a special type of node, called a noise node, to help reduce clutter in the graph. A noise node, shown in Figure 2(b), is an aggregate of features that contain only a small number of data items. The noise node aggregation is performed on a per feature set basis and is colored by the parent feature set. The noise nodes are given a special type of rendering so the user does not mistake it for a normal feature.

2) Edges: The edges of the knowledge graph represent the intersection of data items between features. In terms of two different photo database searches, intersection means that the features contain at least one photo in common. For standard partial intersection, i.e. where $A \cap B \neq \emptyset$ and $A \cap B \neq A$ for $|A| \leq |B|$, we use a straight edge between nodes, as shown in in Figure 3(a). The edge size describes the number of items in the intersection and the color and brightness describe what percentage of each node is contained in that intersection.

Since part of the knowledge graph's power is the ability to correlate multiple result subsets simultaneously, we also generate hyperedges, such as the one shown in Figure 3(c). A hyperedge shows the multi-intersection among a group of nodes in the same fashion as a binary edge. Whenever a group of three or more feature nodes share data items then there exists a subset of data contained in all those features. If this exists, then a hyperedge can be drawn as a star structure by having an outgoing edge from every node meet at a centralized crux, which we define as the centroid of the intersecting nodes. The same color scheme is used here as with binary edges, and thus, the hyperedge shows the overlap of many sets at a time.

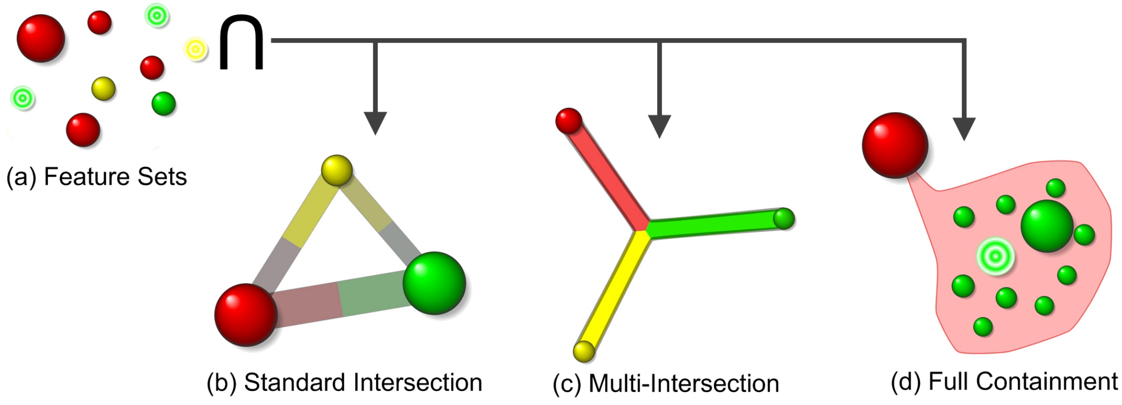

Fig. 3. When intersecting multiple features (a), partial intersections are shown as straight edges, (b) where the color shows the percentage of containment for each node. The brighter the color, the more of that node is contained in that intersection. When multiple features simultaneously share data, a multi-edge is used as in (c). When features are subsets of another feature, they are aggregated into a one-way bubble edge (d).

In the case of total containment, i.e. where $A \cap B = A$ for $|A| \leq |B|$, we use a specialized type of edge, called an edge bubble, that groups all such nodes into a single collection of contained nodes. This helps to reduce visual edge clutter in the knowledge graph by aggregating nodes with a commonality and eliminating unnecessary edge rendering. For every node A_i that is completely contained in a node B, the nodes A_i are clustered together and surrounded by a polygonal boundary, as shown in in Figure 3(d). This polygon boundary then connects to the node B just like an edge and is filled with the color of node B's set.

3) Hypergraph Layout: In order to better utilize screen space, the graph is positioned using a force directed layout [5]. During the layout phase of the visualization, however, the algorithm takes certain hidden forces into consideration. To properly handle the contents of an edge bubble, the nodes inside the bubble are conceptually tied together and attached to the larger feature node. This keeps the bubble tightly bundled, which reduces clutter and improves visual appeal. Hyperedges do not influence the layout of nodes in the graph; only the the binary edges, or simple intersections, are used. Since a complete binary subgraph exists for every hyperedge's incident nodes, it is unnecessary to use the hyperedge in the force calculations, regardless of its visibility.

A significant modification to the force directed layout is the addition of interactive magnets, which is similar to a technique that has been applied to multivariate data [22]. The purpose of magnets is to organize a scattered collection of nodes and edges into more meaningful sectors. A magnet exists for each result subset, i.e. for every search query, and can be repositioned by the user. The magnet stays anchored to its set location and attracts all of its corresponding feature nodes toward it, as is shown in Figure 2(d). Since all magnets pull on their respective nodes, the end result is a separation of connection types, e.g. unconnected nodes float around the magnet while nodes with a standard edge to another node will be positioned in the middle of the two feature magnets.

C. Interactive Exploration and Modification

While the force directed, magnet layout reduces overlap, other tools are required for exploring the graph in detail and reducing clutter. One of the simplest ways of modifying the knowledge graph is by moving through the hierarchy levels, revealing more detailed feature sets at lower and lower levels. In order to help the user keep track of information between transitions, such as adding a new result set or changing the hierarchy level, we provide animations that lead the user from one stage to the next. When adding a new set, the nodes are faded into the existing knowledge graph gradually with the layout only being modified slightly from the previous one. These types of animated transitions are used throughout to provide visual feedback of changes being made, similar to the work done by [3].

Besides being able to move from one hierarchy level to another to see each feature set, it is occasionally advantageous for a user to see a single node's children without completely changing the view. In this case, a user may select a node from the current feature set and expand its contents much like a balloon graph. The children nodes become surrounded by an expanded circle to show that they are on a lower level than the rest of the feature set. The edges of the children become visible as they connect to higher level nodes outside of the balloon.

Additional interface options are provided for filtering nodes and edges. Cardinality sliders can hide nodes or edges that do not fall within the specified range. The layout updates interactively as items are hidden. We also allow individual edge types to be selectively made invisible without disrupting the layout positioning. For example, the user may select only tri-edges, which represent the intersection of three sets, or quad-edges to be rendered while hiding all others. This allows some fine grain control to generate the exact image the user would like to see.

A more powerful interactive tool is the edge toggler. When viewing the knowledge graph, it may not be useful to see intersections between certain features sets and not others.

Thus, we provide a simple user interface that allows the user to turn on or off edges between any pair of feature sets. The unwanted edges become invisible, and the force directed layout will adjust node positions accordingly, as if those nodes no longer attract. This would be analogous to a Euler diagram choosing not to overlap circles between certain sets.

The system also allows a user to see the original data represented by each feature node through mouse selection. When a node or edge is picked, the original data is displayed in a side window. The default view is a table of entries and specialized views, such as a photo viewer for flickr data, can be written for specific types of data.

IV. Results

Now we will show the overlap of search results from Flickr data and social network data. While the method of obtaining and clustering the results is not a topic of this paper, we created a simple test application that uses a SQL database for queries and simple categorization that is coded specifically for each different dataset. It is possible to find all possible types of intersections manually using SQL queries, but knowledge graphs present a complete view of all intersections simultaneously and provide a suite of interactive methods that let the user explore the hierarchy in context while creating a custom view of the data.

A. Flickr

Our Flickr database contains over 13 million photo details pulled from Flickr prior to 2009. Each photo entry contains 31 values, including user, timestamp, view count, location, title, and technical specifications, plus any number of possible tags. Since the last images we collected were at the end of the 2008 U.S. presidential elections, we performed our case study to examine the popularity of the candidates over time based on their appearance in photo tags.

In Figure 4 we begin with five tag searches, which include the president and vice-president candidates and the tag 'election'. The first level is, of course, one node per result. We then create the feature hierarchy by dividing nodes by date of photo, with the second level being year, the third level being month, etc. Level 1 of Figure 4 shows how some of the interactive tools are used to discover high-level aspects of the data. First, the nodes appear fully connected, and with edge hiding, we show a hyperedge representing photos containing all five tags. Second, by ballooning out the nodes and using the edge toggler, we can see just how the tag 'election' relates to every other tag based on year. In this example, 2008 is the largest year for all of the feature sets, and it is the only year that the the vice-president candidates appear with the tag. In addition, there are some photos from the years 2006 and 2007 that reference either John McCain or Barack Obama.

We transition to the next hierarchy level, which reveals features sets separated by year. We remove the 'election' tag to focus on just the candidates. In 2008, Barack Obama is the most popular tag of the group, which can be seen by comparing feature node sizes, so we decide to expand that

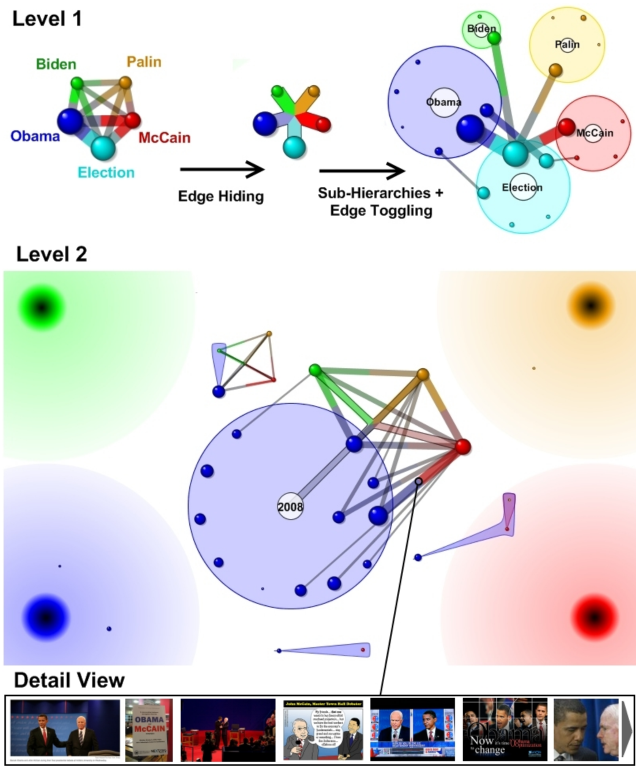

Fig. 4. The above example shows feature hierarchies from a Flickr database. Each hierarchy represents a tag pertaining to the 2008 US presidential elections clustered by time. Using several interaction techniques, the first level reveals 2008 was the most popular for photographing all candidates, and the second level shows how Barack Obama became more prominent as the November election approached. A detailed view of one of the edges between Obama and McCain is shown at the bottom.

node to reveal it's sub-hierarchy. If we mouse over the nodes to reveal the cluster labels, we see that the most popular month is November, the same month as the election. Also, Obama and McCain began appearing together before either of the vice-president candidates came into the picture. A detail view of images is seen below the graph by selecting the edge connecting Obama and McCain during the month of October. Overall, this exploration of the knowledge graph reveals the rise of important figures in the media during the long election process. Without knowledge graphs, the same conclusions might require multiple searches or visualizations of the data in order the represent the same hierarchy and overlaps.

B. Orkut Network

Our social network dataset, which comes from the Orkut network (a Google product), contains over 15,000 anonymized user profiles and over 60,000 friend connections. The feature hierarchies in this case are generated using community structure clustering methods commonly found in social network studies. Therefore, each node represents a cluster of closely connected friends. In the example shown in Figure 5, instead of taking queries intended to generate highly correlated groups of people, the queries were chosen randomly, without considering the others. The feature hierarchies are as follows: people

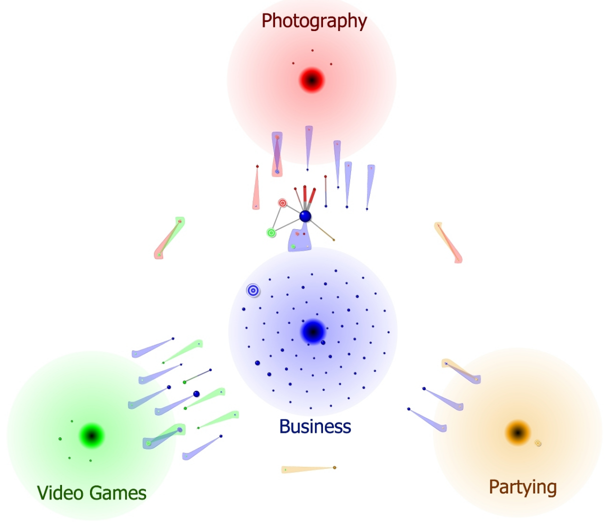

Fig. 5. In this figure, four unrelated search queries are combined into a knowledge graph. The sets represent the interests of Orkut users, divided into photography, video games, business, and partying. Even though no overlap was intended, some users were found having eclectic listings of activities.

interested in cameras or photography (red), people interested in video or computer games (green), people over the age of 20 on Orkut for business (blue), and people interested in partying or hanging out (orange).

The knowledge graph here shows where these very different feature hierarchies have commonalities. Between each feature set, there exists a small amount of intersection that indicates a combination of those interests. In the center, a large community of business minded people can be seen sharing interests from each of the other feature sets. Since these are not full containment or subgraph cliques, these are minorities from the larger feature node. A social network researcher may be able to discover individuals who appear in many popular, unrelated searches, which could then lead to an understanding of some small outlying connection among them all.

V. Conclusion

This paper has introduced knowledge graphs, a visualization of overlapping feature sets. Using a modified node-link diagram representation where nodes are features and hyperedges are intersections, our system, KnowHow, provides a high level view of feature hierarchies from a wide range of different datasets. Unlike other set relation diagrams, knowledge graphs focus on sets that have been hierarchically clustered. KnowHow utilizes various aggregation, abstraction, and interaction techniques to help reduce the complexity of the feature space and allow for feature exploration. We have applied this system to a photo database and a social network. Without modification, KnowHow is able to abstract different types of data into the same feature space, which shows that the knowledge graph representation is versatile enough to be applied to many areas.

It is clear from our evaluation of the system that being able to comprehend and navigate the knowledge graph requires the user to become familiar with the new abstract view of their features. Linking the feature nodes directly to the original data helps in this regard, though, as it maps the abstract features back to the actual features. As our work on feature space visualization continues, we are prepared to perform user studies to test the ability of users in using knowledge graphs to answer questions about features and its effectiveness as an interface for exploring the original data. In addition, we plan to study enhanced visual representations that may be more intuitive at conveying the same information or are capable of simplifying the information.

References

[1] S. Carpendale and C. Collins. Vislink: Revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1192–1199, 2007.

[2] R. D. Chiara and A. Fish. Eulerview: a non-hierarchical visualization component. *Symposium on Visual Languages and Human Centric Computing*, pages 145–152, 2007.

[3] P. Eades and M. L. Huang. Navigating clustered graphs using force-directed methods. *Journal of Graph Algorithms and Applications*, 4(3):157–181, 2000.

[4] J. Flower, A. Fish, and J. Howse. Euler diagram generation. *Journal of Visual Languages and Computing*, 19(6):675–694, 2008.

[5] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences USA*, 99(12):7821–7826, June 2002.

[6] D. Harel. On visual formalisms. *Commun. ACM*, 31(5):514–530, 1988.

[7] D. W. Henderson. Venn diagrams for more than four classes. *American Mathematical Monthly*, 70:424–426, 1963.

[8] T. K. Ho. Interactive tools for pattern discovery. In *International Conference on Pattern Recognition*, pages 509–512, 2004.

[9] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, September 2006.

[10] D. Holten and J. J. van Wijk. Visual comparison of hierarchically organized data. *Computer Graphics Forum*, 27(3):1192–1199, 2008.

[11] D. S. Johnson and H. O. Pollak. Hypergraph planarity and the complexity of drawing venn diagrams. *Journal of Graph Theory*, 11(3):309–325, 1987.

[12] H. Kestler, A. Muller, J. Kraus, M. Buchholz, T. Gress, H. Liu, D. Kane, B. Zeeberg, and J. Weinstein. Vennmaster: Area-proportional euler diagrams for functional go analysis of microarrays. *BMC Bioinformatics*, 9(1), 2008.

[13] P. Neumann, S. Schlechtweg, and M. S. T. Carpendale. Arctrees: Visualizing relations in hierarchical data. In *EuroVis*, pages 53–60, 2005.

[14] H. Omote and K. Sugiyama. Method for visualizing complicated structures based on unified simplification strategy. *IEICE - Transactions on Information and Systems*, E90-D(10):1649–1656, 2007.

[15] G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins. Polyarchy visualization: visualizing multiple intersecting hierarchies. In *Conference on Human Factors in Computing Systems*, pages 423–430, 2002.

[16] P. Rodgers, P. Mutton, and J. Flower. Dynamic euler diagram drawing. In *Symposium on Visual Languages and Human Centric Computing*, pages 147–156, 2004.

[17] F. Ruskey, C. D. Savage, and S. Wagon. The search for simple symmetric venn diagrams. *Notices of the American Mathematical Society*, 53(11):1304–1311, 2006.

[18] P. Simonetto and D. Auber. Visualise undrawable euler diagrams. In *Intl. Conference on Information Visualisation*, pages 594–599, 2008.

[19] A. Spoerri. Infocrystal: a visual tool for information retrieval. In *Visualization Conference*, pages 150–157, 1993.

[20] K. Sugiyama and K. Misue. Visualization of structural information: Automatic drawing of compound digraphs. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 21, pages 876–892, 1991.

[21] A. Verroust and M.-L. Viaud. Ensuring the drawability of extended euler diagrams for up to 8 sets. In *Diagrams*, pages 128–141, 2004.

[22] J. S. Yi, R. Melton, J. Stasko, and J. A. Jacko. Dust magnet: multivariate information visualization using a magnet metaphor. *Information Visualization*, 4(4):239–256, 2005.