GopherCluster: Document Visualization in the Third Dimension

Paul Lindner
lindner@boombox.micro.umn.edu
GopherCon '95

In it's present form Gopher is a one-dimensional model; a simple linear list. By using three dimensions to display the same data we can visualize the innate relationships between documents. One way of visualizing a database of text documents is by putting similar documents close together in what we call a Cluster. This paper describes a new system, called "GopherCluster". This system attempts to build document clusters automatically by using an iterative force function.

Introduction

Until recently information systems such as Gopher and the World-Wide-Web have been confined to linear lists or graphical pages. These systems use real-life metaphors to visualize and organize information. Overall the book metaphor is the most popular. Infosystems routinely support the equivalent of:

- Table of contents
- Indices
- Footnotes
- Quoted references to other material

Recently we've started to see non-book metaphors for organizing information. The Blue-Skies Gopher client allows interactive spatial organizations of information based on links overlaid on a graphic image. Similar software, called clickable image maps, exists in World Wide Web software as well.

Still, for most infosystems we're still stuck in the book frame of mind. Most lists of documents are simple linear listings of references; most infosystem indices mimic their real-world counterpart.

The advent of three dimensional software that runs on inexpensive hardware should change this. With a third dimension it's possible to visualize information in ways never dreamed possible. Three dimensional Systems offer a three-fold increase in the number of dimensions that can be simultaneously represented.

Consider the Gopher information system. The following are the traits of a Gopher reference:

- Name of item
- Location in list
- Type of item
- Abstract for item
- Reference to a remote item.
- Miscellaneous metainformation

The World-Wide-Web adds the following traits to an information system

- Font Size
- Font Style (Bold, Italic, etc.)
- Associated graphic icon

When we move to the third dimension we gain many new traits to describe information.

- Colors
- Size in three dimensions
- Abstract shape representation
- Vertical position and size

- Horizontal position and size
- Depth position and size
- Orientation of object
- Associated text
- Associated sounds and scores
- Location of object relative to other objects

We can map most of these traits to data within the information system. The height of an object can be mapped from the size of the document. The abstract shape can represent the type of the document, or a preview of what the document contains.

A harder problem is mapping the relative location onto data within the information system. The software described in this paper seeks to do just that. Some information systems allow you to find the similarity between two documents. With this data in hand we set out to group the information into distinct separate areas called clusters.

A cluster should be representative of a group of documents that are self similar. For instance a recipe information system could group all the cake recipes into one area and all the meat recipes in another and so on.

The problem with clustering is how to do it efficiently for large sets of documents. We could do it manually. This organization would have the highest quality since human thought would go into the organization of the information. For large document sets we need to look beyond manual organization. A software solution is the most efficient answer. We hope that GopherCluster can fill this need.

Our Clustering Algorithms

Prior Work

We did some literature searches for information on clustering in information systems but came up very short. It is a difficult problem to represent documents in clusters.

Still, some work has been done in this area

- Jussi Karlgren[1] at the Sweden Institute of Computer Science (SICS) has done a lot of work in the area of clustering documents based on user behavior patterns.
- Scatter-Gather is a system that organizes information into different 'classes' that the administrator defines. It was developed at Xerox PARC.
- There was at one time an implementation of clustering for the SMART system. This was one of the reasons we initially looked at it. However that code does not work on the current SMART code base.

In most cases the focus of these studies was more on grouping items together than representing a scene in three dimensions. We're still looking for more prior work in this area.

---

1. See <URL:http//sics.se/~jussi/jussi-karlgren.html>

The social sciences have long done multivariate cluster analysis. This has allowed scientists in these fields to come up with generalizations of their data samples:

- Smokers have a high risk of cancer.
- Children in high income families score higher on standardized tests.
- People who eat bran muffins and sleep on their side are more likely to watch comedies on television.

We found these algorithms and methods interesting, but we found it difficult to apply them to our problem. A number of factors kept us from using standardized clustering algorithms, including:

- A lack of independent variables. Most cluster analysis systems assume you have a large set of data that you want to find clusters for. We only had the similarity between items.
- We have no method of automatically generating the cluster attributes. This generally requires a semantic understanding of the content of the documents. This would require human understanding or a very good expert system.

Most of the algorithms seem focused on grouping the items into clusters, not organizing and mapping them to a three dimensional representation. We want to use the human spatial senses to visualize the clusters, not a computer telling us where the clusters are.

Our Algorithms

Instead we set out on our own with a gut feeling of how clustering could be done. We had the following information:

$P_d[x,y]$ initial location of document d

$S(P_{d1},P_{d2})$ similarity of documents d1 and d2

We can represent the similarities between documents as a matrix like this:

TABLE 1. Matrix of similarity values

xx	d1	d2	d3	d4
d1	1	0.34	0.04	0.55
d2	0.34	1	0.22	0.33
d3	0.04	0.22	1	0.08
d4	0.55	0.33	0.08	1

We'll call this matrix Sim[d1,d2]

We tried a number of approaches to clustering documents, most were unsatisfactory:

- Our first try represented the x-axis as the similarity to the highest ranked document. The y-axis was represented as the similarity to the second highest ranked document. This didn't work well. We ended up with one big clump in most cases.

- Next, we tried representing things in polar coordinates. Less relevant documents were farther away, more relevant ones are closer. Documents similar to the best document were swept out right to left. This really didn't generate clusters either. It was an interesting representation though.

Nether approach generated what we would consider an actual cluster.[1]

Finally, we hit upon a method that seemed to work. We stopped thinking of ways to generate variables to plot onto the different axis. Instead we transformed our system into a physical model.

We considered each document to be a mass on a two dimensional grid. This allowed us to map the similarity factors to a 'force' between the two entities. The following steps led us to what seemed to be actual clustering.

1. Calculate the initial location of the documents on the grid. We find the square root of the number of documents and spread the documents over a square that size.
2. Calculate the accumulated forces on each document. Each document pulls or pushes other documents based on their closeness.
3. Move the documents to their new locations.
4. Check to see if a significant portion of the documents are close together. Go back to step 2 if this hasn't happened.

We used the following force functions to move the documents around given D= the number of documents:

Force functions used in GopherCluster **(EQ 1)**

$$P'_d = \sum_{i = [0, D]} \sum_{\substack{j = [0, D] \\ i \neq j}} \frac{P_j - P_i}{\left|P_j - P_i\right|} S(i, j) \qquad \text{given distance} > 2.0$$

$$P'_d = \sum_{i = [0, D]} \sum_{\substack{j = [0, D] \\ i \neq j}} -\left(\left(P_j - P_i\right)\left(1 - \left|P_j - P_i\right|\right)\right) \qquad \text{given distance} < 1.0$$

The final location of the point is the sum of these two force equations. The first sums the attractive force between documents. The second function repels documents that get very, very close together so they're exactly one unit away.

We stop the system when we reach a point where 50% of the documents are within two units of each other. If we didn't do this, the system would congeal into one large clump.

---

[1] Unless you consider one big clump of documents a cluster.

The Implementation

We used the following tools to make the clustering searches happen

- The SMART Text Retrieval system
- The GopherVR software
- The Unix Gopher Server
- Two perl scripts

SMART

The backbone of the whole clustering system is the SMART text retrieval system. SMART was developed at Cornell University to provide a solid framework for work in information retrieval. You can get more information about SMART by selecting the following URL:

> <URL:ftp:ftp.cs.cornell.edu/pub/smart/>