Project Specification

Scatter/Gather Browser: a Visual Web Interface for Text Navigation and Search

Alex Berry, Sujit Gadkari, and Weimao Ke
School of Library and Information Science
Indiana University, Bloomington
1320 E. 10th Street, LI 011
Bloomington, Indiana 47405-3907
{berry3,sgadkari,wke}@indiana.edu

1 Introduction

Effective and efficient browsing methods for large text collections have been widely examined in recent years. Among existing implementations of various browsing methods, Scatter/Gather browsing is well known for its ease to use and effectiveness in situations where it is difficult to precisely specify a query (Cutting, Karger, Pedersen, and Tukey 1992; Hearst and Pedersen 1996). It combines search and interactive navigation by gathering and reclustering user-selected clusters (e.g. represented by levels of keywords).

Scatter/Gather browsing method was first proposed by Cutting, Karger, Pedersen, and Tukey (1992). In each iteration of this browsing method, the system scatters the dataset into a small number of clusters/groups, and presents short summaries of them to the user. The user can select one or more of the groups for future study. The selected groups are then gathered together and clustered again using the same clustering algorithm. With each successive iteration the groups become smaller and more focused. Iterations in this method can help users refine their queries and find the desired information from a large data collection.

This project aims to implement a Scatter-Gather browser, a dynamic visualization for text navigation/search. Using visualization techniques, this browser will help users refine their search queries and narrow down search results interactively and visually. We are going to constraint ourselves to a smaller text corpus for proof of concept. We will modularize it to be able to attach to any text collections in the future.

2 Design Considerations

Figure 1 shows a prototype of the Scatter/Gather browser. Note that the implementation of the browser will be purely HTML-based and reside within a Web browser.

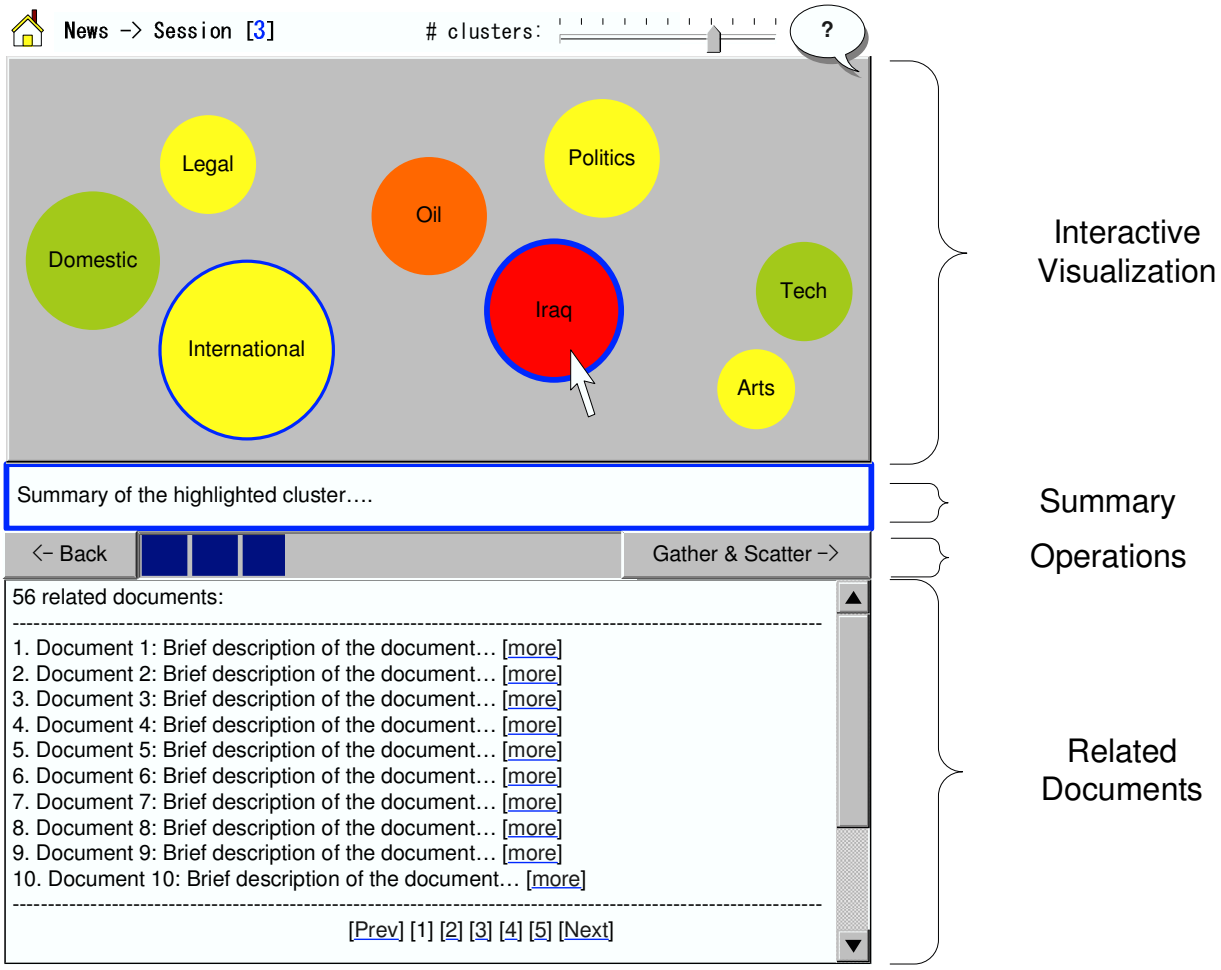

Figure 1: Scatter/Gather browser layout

As shown in Figure 1, the browser looks very different from traditional web sites. It does not have a navigation bar, which is commonly used by many others. When a user comes to the site, the browser shows the initial/top clusters of its text corpus. On the visualization, each node represents a cluster. Node size represents the number of documents in each cluster. Node color denotes the popularity of a cluster. A warmer color indicates a hotter topic. Positions of the nodes are based on their similarities–the layout algorithm tries to place similar clusters close together and disimilar clusters far apart on the 2-D space.

When the cursor is moved over a node, the summary panel shows a more informative description of the cluster. When the user clicks a node, the cluster is selected and highlighted with blue border color. The user can click again to deselect the item. Once clusters are chosen, the user can click the "Gather & Scatter" button to produce new

clusters for the selected documents. In case the user realizes what she has chosen is not satisfactory, she can click the "Back" button to revoke the last reclustering and go back to the previous session. The user can also move the slider to change the number of clusters to be produced. The default value of this is 7 while the maximum is 15.

3 Information Flow

Figure 2 shows an information flow of the scatter/gather system. It begins with preprocessing a text collection: term extraction, stopword removal, stemming, term weighting, and indexing. The processed document/term matrix is stored in a database, which is then used by the online browser for clustering and document retrieval. When a user comes in, the system clusters the whole collection and presents the results. The user can select favored clusters and gather & scatter them to produce new clusters. This continues to refine the search results until the user is satisfied.

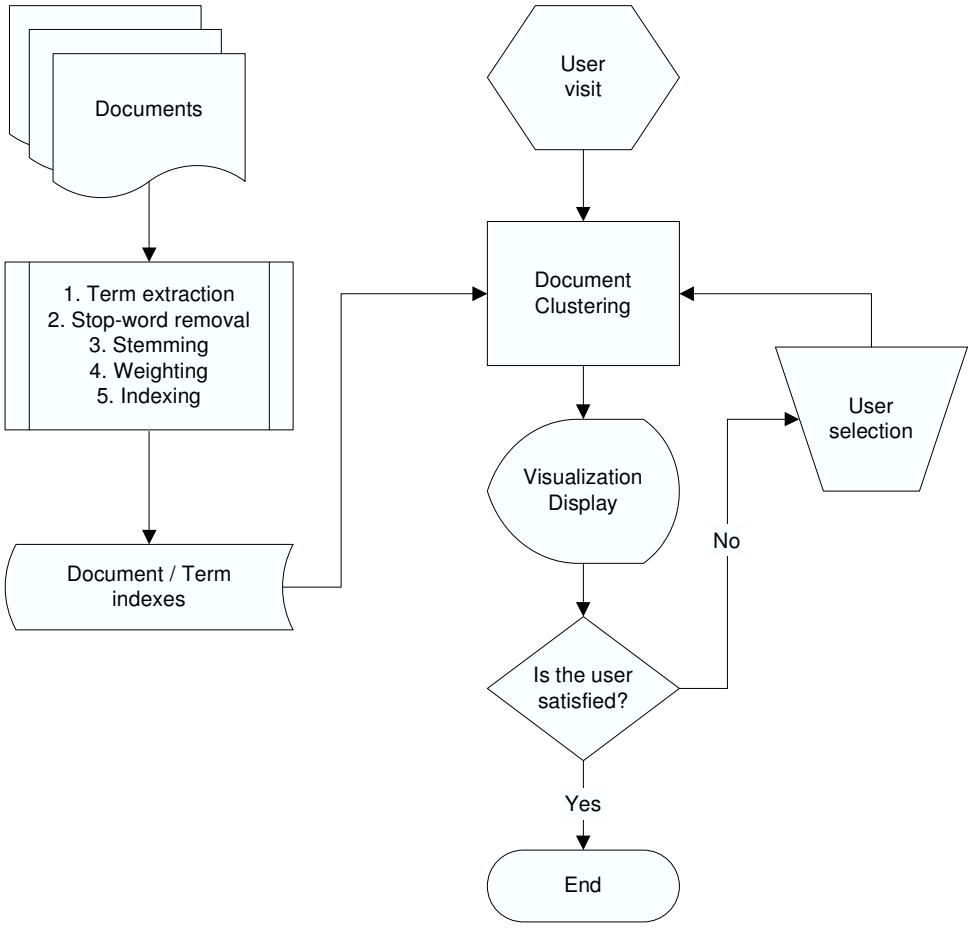

Figure 2: System Flowchart

3.1 Document Representation

For document representation, we use the widely used Vector-Space Model (Salton, Wong, and Yang 1975) to construct document-term vectors. Feature selection produces a thesaurus for a document collection. This thesaurus is then used to represent each document using the TF*IDF (term frequency * inverse document frequency) weighting scheme. A document is then converted to a numerical vector where the i^{th} item is computed by:

$$W_i = T_i * log(N/n_i) \tag{1}$$

Where T_i is the frequency of the i^{th} term of the thesaurus in the new document, N is the total number of documents in the representative document set, and n_i is the number of documents in the representative document set containing the i_{th} term of the thesaurus. The TF*IDF is a well-known and effective technique for term weighting (Baeza-Yates and Ribeiro-Neto 2004).

3.2 Offline Clustering

To cluster documents, we use a similarity measure based on term vectors. Document-term vectors are produced by the TF*IDF representation scheme described above. Then a similarity measure, called Cosine Similarity Coefficient (Korfhage 1997; Baeza-Yates and Ribeiro-Neto 2004), is used to compute the cosine of an angle between two vectors. Given two non-null vectors $X = [x_1, .., x_t]^T$ and $Y = [y_1, .., y_t]^T$, their cosine similarity is computed by:

$$\sum_{i=1}^{t} x_i y_i / \sqrt{(\sum_{i=1}^{t} x_i^2)(\sum_{i=1}^{t} y_i^2)} \tag{2}$$

Using the Cosine Similarity measure, an agglomerative hierarchical clustering algorithm compares the documents and produces a cluster dendrogram for the document collection, illustrated in Figure 3. The algorithm first considers each document in the collection as a separate cluster. Then at each step, two clusters with the highest similarity are agglomerated. After all of the document clusters are agglomerated into one cluster, the final hierarchical cluster structure, which is called dendrogram, is formed.

3.3 Online Reclustering

Given the precomputed cluster dendrogram, online reclustering (scattering) is to browse the hierarchy and to retrieve a number of sub-clusters needed by the user. We use the example in Figure 3 to elaborate on the process.

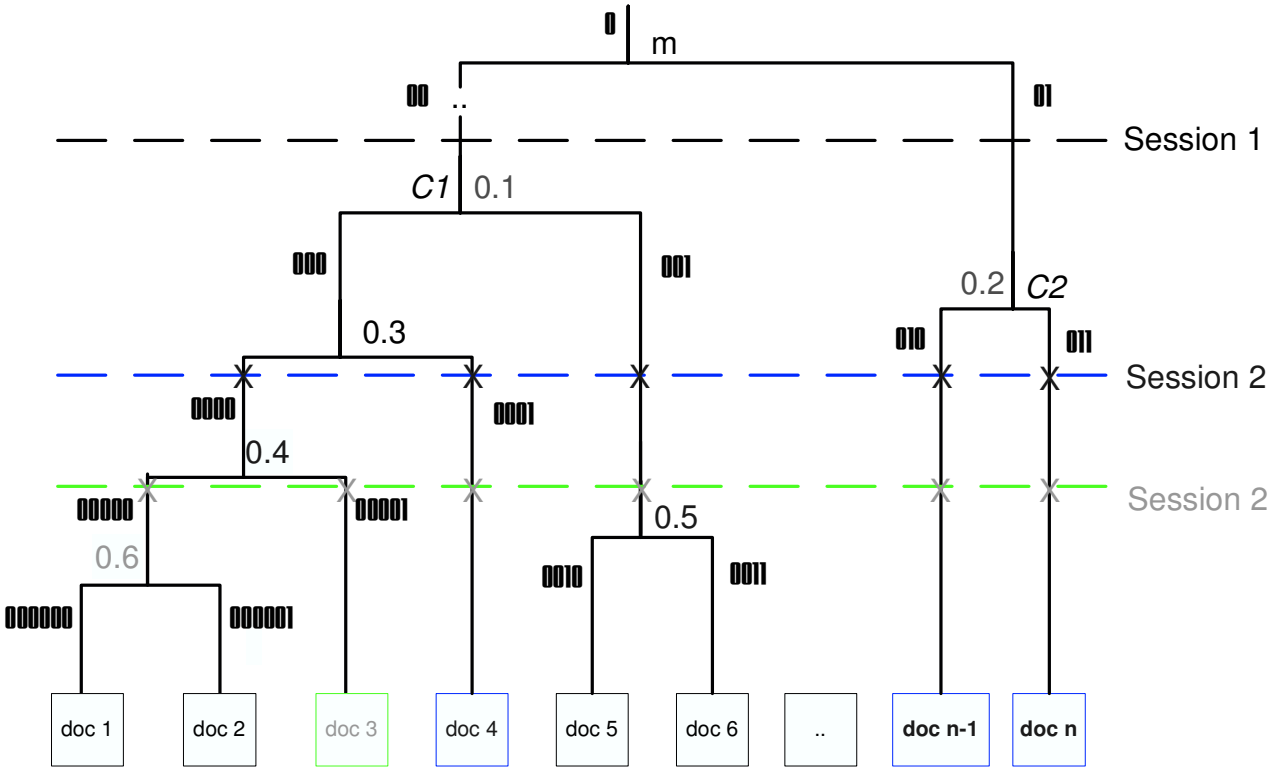

Figure 3: Dendrogram

In the first iteration/session, suppose the user chooses two initial clusters, C1 and C2. Then she clicks the Gather-and-Scatter button. What the system does is just to move a horizontal line down the dendrogram to find more clusters, i.e. descendant clusters of the chosen ones. In this case, the system is able to retrieve 5 clusters when it reaches the blue dash-line. If, for instance, the user needs 6 clusters, the green dash-line works. We elaborate on the *tree_id* (e.g. '00','001', etc. on Figure 3) and *height* (e.g. 0.1, 0.2, etc. on Figure 3) in Section 4.

4 Database and Data Structure

For efficient online retrieval of clusters, the precomputed dendrogram will be served in the database. Figure 4 shows the proposed database tables and their entity relationships. Note [*clusters*].[*height*] denotes the vertical level of a cluster where its subclusters meet/merges/agglomerate. The larger the value, the closer a cluster is to the bottom and more distant from the top level. The [*clusters*].[*tree_id*] is a string representing parental/descendant relations. For instance, the very top-level cluster is '0'. Since this is a binary tree, it has two children, '00' and '01'. And '00' has children '000' and '001' while '01' has '010' and '011'. In this way, the whole dendrogram can be mapped to the

[*clusters*] table. Another example of this representation is also shown in Figure 3–see those '00', '001' values.

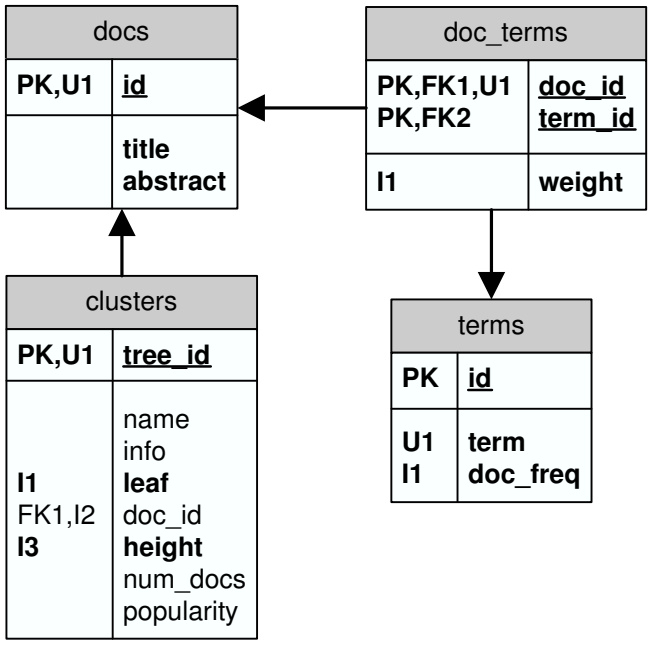

Figure 4: Database Entities and Relationships

Using the [*clusters*].[*tree_id*] and [*clusters*].[*height*], it is very straightforward to rescatter the chosen clusters and to retrieve a new level of clusters with SQL. Suppose the user has chosen clusters '000' and '0110', to reproduce 7 new clusters is to iterate the following *height* values and cut the hierarchical binary tree until enough number of clusters are retrieved:

```
1: SELECT height FROM clusters
2: WHERE tree_id like '000%' or tree_id like '0110%'
3: ORDER BY height ASC
4: LIMIT [NO. ITERATIONS];
```

5 Project Members and Roles

- Alex Berry (berry3@indiana.edu): the lead programmer on CGI and (D)HTML interface.
- Sujit Gadkari (sgadkari@indiana.edu): a user representative and the lead HCI designer.
- Weimao Ke (wke@indiana.edu): the lead programmer on IR functions and data structure; documentation.

6 Helper Applications

- Graphic design: Photoshop, Ulead Photoimpact.
- HTML/CGI editor: vi, pico, Ultraedit.
- Layout, Flowchart, and Database design: Dia, MS Visio/Word.
- Visualization API: Yahoo UI, Ajax (Asynchronous JavaScript and XML).
- Techniques: Perl/CGI, HTML/DHTML, SQL/PostgreSQL.

7 Text Collection

The text corpus we are going to use with this project is INSPEC, a computer/information science collection of 6,000 text documents. The documents are distributed among the following categories:

No.	Category	# docs		No.	Category	# docs
1	data-mining	400		9	information-retrieval	400
2	data-models	400		10	data-visualization	400
3	electronic-publishing	400		11	knowledge-representation	400
4	full-text-databases	400		12	medical-information-systems	400
5	indexing	400		13	multi-agent-systems	400
6	information-dissemination	400		14	page-description-languages	400
7	information-needs	400		15	software-agents	400
8	knowledge-acquisition	400			Total: 6,000 docs	

Table 1: INSPEC Text Collection

8 Project Programs

https://ella.slis.indiana.edu/g/l548s07c/site2/programs.html

References

Baeza-Yates, R. and B. Ribeiro-Neto (2004). *Modern Information Retrieval*. Addison Wesley Longman publishing.

Cutting, D. R., D. Karger, J. O. Pedersen, and J. W. Tukey (1992). Scatter/gather: A cluster-based approach to browsing large document collections. In *The 15th Annual ACM-SIGIR*, pp. 318–329.

Hearst, M. A. and J. O. Pedersen (1996). Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 76–84. ACM Press.

Korfhage, R. R. (1997). *Information Storage and Retrieval*. Wiley Computer Pub.

Salton, G., A. Wong, and C. S. Yang (1975). A vector space model for automatic indexing. *Commun. ACM 18*(11), 613–620.