Toward Responsive Visualization Services for Scatter/Gather Browsing

Weimao Ke, Javed Mostafa, and Yong Liu
Laboratory of Applied Informatics Research
University of North Carolina at Chapel Hill, 216 Lenoir Drive CB#3360, 100 Manning Hall, Chapel Hill, NC 27599-3360, U.S.A.; Tel: 919- 962-8366, {wke@unc.edu, jm@unc.edu, yonliu@indiana.edu}

As a type of relevance feedback, Scatter/Gather demonstrates an interactive approach to relevance mapping and reinforcement. The Scatter/Gather model, proposed by Cutting, Karger, Pedersen, and Tukey (1992), is well known for its effectiveness in situations where it is difficult to precisely specify a query. However, online clustering on a large data corpus is computationally complex and extremely time consuming. This has prohibited the method's real world application for responsive services. In this paper, we proposed and evaluated a new clustering algorithm called LAIR2, which has linear worst-case time complexity and constant running time average for Scatter/Gather browsing. Our experiment showed when running on a single processor, the LAIR2 online clustering algorithm is several hundred times faster than a classic parallel algorithm running on multiple processors. The efficiency of the LAIR2 algorithm promises real-time Scatter/Gather browsing services. We have implemented an online visualization prototype, namely, LAIR2 Scatter/Gather browser, to demonstrate its utility and usability.

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 of use and effectiveness in situations where it is difficult to precisely specify a query (Cutting et al.., 1992, Hearst & Pedersen, 1996). It combines search and interactive navigation by gathering and reclustering user-selected clusters.

The Scatter/Gather browsing method was first proposed by Cutting et al.. (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.

Since the Scatter/Gather method requires online clustering on a large data corpus, fast clustering algorithms are essential. Two linear time clustering algorithms, namely the Buckshot and the Fractionation, were implemented for the original Scatter/Gather method (Cutting et al.., 1992). Both algorithms have $O(kn)$ time complexity, where k is the number of desired clusters and n the total number of documents. As compared to the Buckshot, the Fractionation algorithm is a little slower but with higher accuracy. Although better than a quadratic time complexity, $O(kn)$ is not fast enough for large document collections. Jensen, Beitzel, Pilotto, Goharian, and Frieder (2002) proposed and evaluated a parallel version of the Buckshot algorithm, which achieved a $O(n\log n)$ time complexity.

Cutting, Karger, and Pedersen (1993) proposed an algorithm that used a precomputed hierarchy of meta-documents for further expansion of selected items and reclustering of the subset. Only dealing with a subset of M meta-documents in each iteration, the algorithm achieved constant interaction-time for Scatter/Gather browsing. However, the reclustering process is not efficient enough for real time interaction because M cannot be too small ($M >> k$, the number of clusters needed). On the other hand, by summarizing descendant documents, meta-documents might be too large to be reclustered efficiently, or too small to be accurately representative.

Research Focus and Objectives

Our research focuses on the design and evaluation of a new algorithm for responsive online clustering in Scatter/Gather. The new algorithm takes advantage of a precomputed hierarchy but does not rely on meta-documents for reclustering. We elaborate on our algorithm below and compare it to the approach proposed by Cutting et al.. (1993).

The Proposed LAIR2 Algorithm

We present an new clustering algorithm called LAIR2, which can greatly improve the response time of Scatter/Gather browsing sessions. The algorithm is composed of two phases. In the offline phase, a cluster hierarchy is generated using traditional hierarchical clustering algorithms. Later in the online phase, drawing on the previously generated hierarchy, the online LAIR2 algorithm is used to cluster user selected data items in almost constant time.

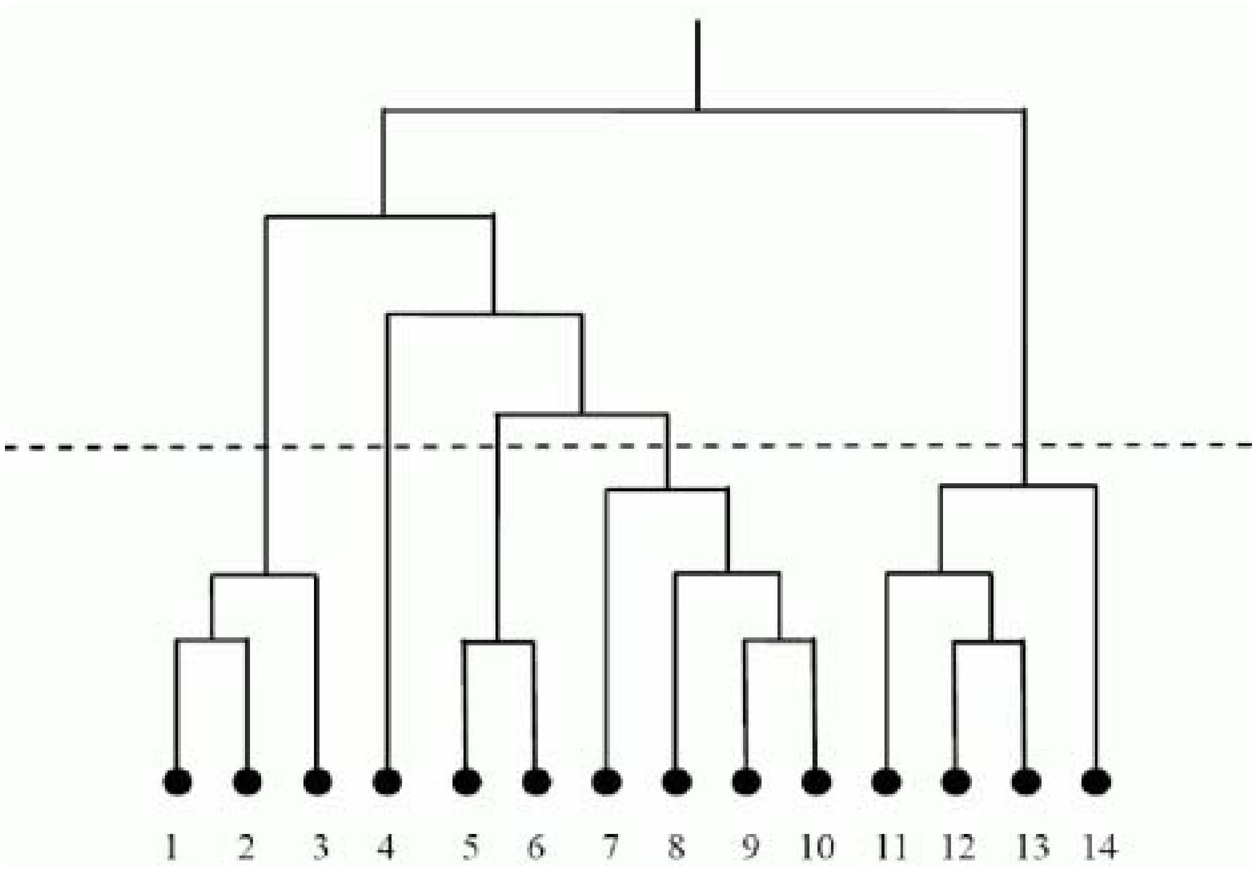


Figure 1: A dendrogram shows how the clusters are agglomerated hierarchically. By cutting the tree at different heights, different number of clusters can be generated. For example, the dashed line in this figure generates five clusters from the dendrogram

In the first phase of the LAIR2 clustering algorithm, an arbitrary agglomerative (or divisive) hierarchical algorithm can be used to construct the dendrogram. The result is represented by a sequence of the agglomerated pairs of data points. Table 1 shows a possible agglomeration sequence of the dendrogram shown in Figure 1.

Table 1: A possible agglomeration sequence of a dendrogram

Seq. No.	Cluster 1	Cluster 2	Similarity	New Cluster No.
1	1	2	0.94	n+1

2	12	13	0.89	n+2
3	5	6	0.70	n+3
4	9	10	0.69	n+4
…	…	…	…	…
n-1	2*n-5	2*n-2	0.21	2*n-1

In each iteration for online reclustering, we decide upon the initial *k* centroids by making use of the agglomeration sequence constructed in the first phase. Suppose the desired number of the clusters is *k* and the number of the clusters in the user selected subset is *k*', and obviously there exists *k*'<*k*. Now the problem is transformed to finding k centroids of the data points which are previously clustered into *k*' groups. Instead of calculating the *k* centroids from nothing, we make use of the previous knowledge, the agglomeration sequence table.

Since we already have *k*' centroids in the current working data collection, we just need to find more centroids to make the total number of centroids equal to *k*. To achieve this, we split the current *k*' clusters according to the precomputed dendrogram in a top-down manner. We scan the dendrogram from the top (or the Table 1 from the bottom), skipping those cluster pairs which have at least one cluster out of the current working data collection. After the first cluster pair whose data points are all in the working collection is found, we split it by removing this entry and adding its two subclusters to appropriate positions in the table. This process is repeated until *k*–*k*' clusters have been split, which means *k* centroids have been identified for the current data collection. The updated agglomeration sequence table is kept for later use in the next Scatter/Gather iteration. The whole process is also illustrated in Figure 2 (a), (b), and (c).

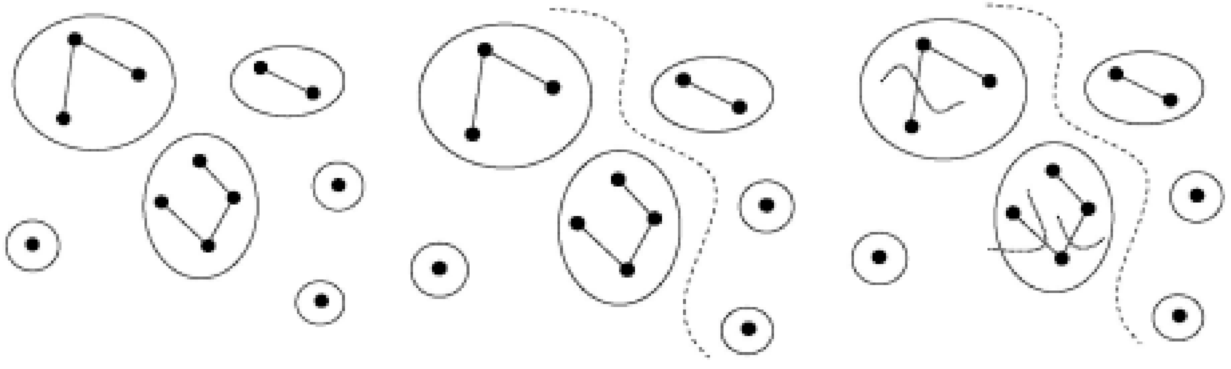

(a) Initial clusters (b) User selection (c) Splitting

Figure 2: Reclustering for online Scatter/Gather

Since the size of the agglomeration sequence table is *n*–1, the worst case time of the split process is *n*+*k*–*k*'–2, which is *O*(*n*). In most cases, the split process will stop in *c*(*k*–*k*') steps, where *c* is a constant related to k and k’. So this algorithm has a constant time complexity (Liu, Mostafa, & Ke, 2007).

Discussion

The LAIR2 online clustering algorithm is potentially faster than the original Buckshot algorithm, which has *O*(*kn*) run time. Another advantage of this algorithm is that it has the same clustering accuracy as the common hierarchical clustering algorithms, which normally require quadratic running time. Compared to hierarchical clustering algorithms, the buckshot algorithm generally has much lower accuracy since it only works on a small random subset to calculate the initial centroids. For the offline phase, incremental hierarchical clustering algorithms may be used to do periodic update (Sahoo, Callan, Krishnan, Duncan, & Padman, 2006, Can, 1993).

Our algorithm looks similar to the one proposed by Cutting et al.. (1993). They both use a precomputed hierarchy for rescattering and achieve constant interaction-time. One might question the value of the new algorithm here because of the similarities. Nonetheless, there are several essential differences. Firstly, although both are of constant interaction-time, our approach simply traverses the hierarchy and expand the selected clusters without online reclustering. This further improves the online interaction efficiency, essential to a system that provides responsive services.

Secondly, our approach remains flexible for a user to select any clusters in each iteration. Although Cutting et al.. (1993) observed that approaches of this kind are too "restrictive" and can serve "only one cluster" at each presentation, this is not necessarily the case. Whereas Cutting et al.. (1993) focus on coarse-grained patterns of local subsets by reclustering meta-documents, our approach maintains a global view of the local ones and reasonably ignores local reclustering. No evidence has shown that local reclustering produces more relevant results to the users. Previous research has supported the usefulness of traversing a hierarchy without reclustering. Crouch, Crouch, and Andreas (1989) built an interactive browser based on a cluster hierarchy of a hypertext collection, which was revealed to be sufficiently comprehensive and flexible enough to support a variety of user searches.

Efficiency Evaluation

Setup

The experiments for the sequential and parallel Scatter/Gather clustering algorithms were conducted on research SP cluster at the Research and Academic Computing Center of Indiana University. The research SP cluster is a distributed-memory system consisting of 144 nodes with a total of 646 processors. Each node on the Research SP runs an AIX system. Experiments were conducted on idle nodes of the research SP. All the sequential and parallel clustering algorithms tested in our experiments were written in Java, using JDK 1.4 and mpiJava library, a wrapper for native IBM MPI interface. Experiments were conducted on a cancer dataset and the 2005 Medical Subject Heading (MeSH) list downloadable from the NCBI PubMed (Lipscomb, 2000). The cancer dataset consists of 51,783 records containing title and abstracts.

Experiment Process and Results

In our experiments we examined both a parallel version of the Buckshot algorithm proposed in (Jensen et al.., 2002) and the LAIR2 clustering algorithms in multiple Scatter/Gather browsing sessions. In each session, both clustering algorithms were used to do online clustering in 10 successive iterations. In each iteration of the Scatter/Gather browsing session, half of the previously generated clusters were randomly selected as the base set on which a further clustering were conducted. For the parallel Buckshot algorithm, 8 processors were used to generate clusters in parallel. For LAIR2 clustering, only one processor was used.

Table 2: Clustering time for Scatter/Gather browsing (*k*=32 clusters)

Time (s) Iteration	Parallel Backshot (8 processors)	LAIR2 (1 processor)
Iteration 1	2.642×10^{1}	3.533×10^{-4}
Iteration 2	1.379×10^{1}	1.813×10^{-4}
Iteration 3	9.177×10^{0}	2.329×10^{-4}
Iteration 4	3.596×10^{0}	6.715×10^{-4}
Iteration 5	2.563×10^{0}	9.348×10^{-4}
Iteration 6	1.367×10^{0}	2.530×10^{-3}
Iteration 7	1.036×10^{0}	1.422×10^{-2}
Iteration 8	6.110×10^{-1}	9.008×10^{-3}

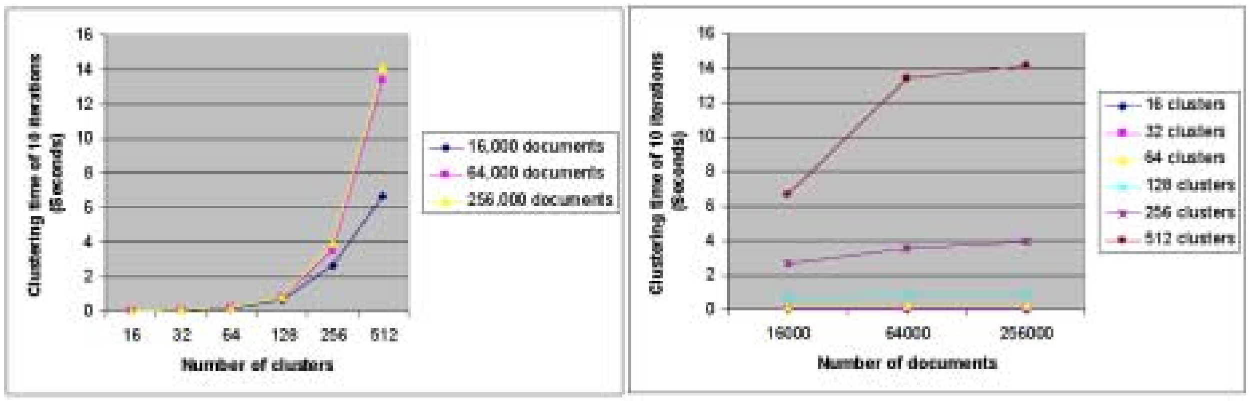
Iteration 9	4.525×10^{-1}	1.984×10^{-2}
Iteration 10	3.406×10^{-1}	3.334×10^{-2}
Average	5.936×10^{0}	8.134×10^{-3}

Table 2 shows the clustering time of each iteration in a Scatter/Gather browsing session using both two clustering algorithms. In this test, 32 clusters were generated in each iteration. After one clustering iteration was done, 16 out of the 32 clusters were randomly picked to be the base of the next iteration, which simulated user selection. From this table we can see that the average response time in a Scatter/Gather browsing session using parallel Buckshot algorithm is around 6 seconds, which is a little slow for interaction time. When using the LAIR2 clustering algorithm, the average response time of a Scatter/Gather browsing session was reduced to several milliseconds. As it is shown in the table, the clustering speed of LAIR2 running on a single processor is several hundreds times faster than the parallel Buckshot algorithm running on 8 processors.

Table 3: Clustering time for Scatter/Gather browsing (*k*=64 clusters)

Time (s) Iteration	Parallel Backshot (8 processors)	LAIR2 (1 processor)
Iteration 1	4.905×10^{1}	1.012×10^{-3}
Iteration 2	2.585×10^{1}	1.059×10^{-3}
Iteration 3	1.446×10^{1}	1.193×10^{-2}
Iteration 4	9.632×10^{0}	1.942×10^{-3}
Iteration 5	2.343×10^{0}	4.030×10^{-3}
Iteration 6	1.231×10^{0}	1.776×10^{-2}
Iteration 7	1.075×10^{0}	2.261×10^{-2}
Iteration 8	5.107×10^{-1}	3.789×10^{-2}
Iteration 9	3.409×10^{-1}	4.072×10^{-2}
Iteration 10	5.364×10^{-1}	5.668×10^{-2}
Average	1.165×10^{1}	1.957×10^{-2}

Table 3 shows the clustering time when the number of target clusters were set to 64. When the parallel Buckshot algorithm was used, the average response time in a Scatter/Gather browsing session was around 12 seconds, which is almost unacceptable for online browsing. Considering the parallel Buckshot algorithm achieves a near linear speedup (Jensen et al.., 2002), the response time will be even worse when using less processors. However, the average response time of the LAIR2 algorithm in this case is 20 milliseconds, which is still quite satisfying. To examine the scalability of the LAIR2 algorithm, we tested it using various number of documents and target clusters in a simulated dataset. The results are plotted in Figure 3 (a) and (b).



(a) Clustering time in terms of # clusters (*k*) (b) Clustering time in terms of # documents (*n*)

Figure 3: Reclustering for online Scatter/Gather

As Figure 3 (a) shows, when the number of target clusters is set to 256, the average response time of the LAIR2 clustering algorithm is around 0.4 seconds on a dataset of 256,000 documents. Only when the number of target clusters increases to 512, the response time of LAIR2 starts to increase rapidly. However, in real situations, 512 is a extremely huge number. We would reasonably expect a user to request for 10 - 20 clusters in each Scatter/Gather iteration.

Figure 3 (b) shows that the number of target clusters is a dominating factor in the response time. The number of documents to be clustered contributes much less to the time, especially when the number of target clusters is small. This result demonstrates that the LAIR2 clustering algorithm has an almost constant running time. This efficiency improvement is very important to the Scatter/Gather method, which relies heavily on online clustering. It will enable real world applications to provide responsive Scatter/Gather services to real time users.

Scatter/Gather Visualization Prototype

A limited number of user studies have been performed on Scatter/Gather interfaces. Hearst and Pedersen (1996) evaluated Scatter/Gather browsing on the TREC-4 interactive track collection. Specifically, they analyzed how often the subjects chose the cluster with the largest number of relevant documents after issuing the first search and re-clustering the results by means of the Scatter/Gather function. Their study, which involved four graduate students each issuing 13 queries, found that users predominately chose the cluster with the largest number of relevant documents. The shown efficiency of the LAIR2 algorithm will enable real world applications to provide responsive Scatter/Gather services to users, who, according to Hearst and Pedersen (1996), "are able to take advantage of the benefits that clustering can provide." To demonstrate its utility, we have implemented a prototype system called LAIR2 Scatter-Gather browser to serve online users. Using visualization techniques, this browser will help users refine their search and narrow down search results interactively and visually.

System Information Flow

Figure 4 shows an information flow of the Scatter/Gather system. It begins with preprocessing a text collection: term extraction, stopword removal, 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 top-level results (7 clusters by default). 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.

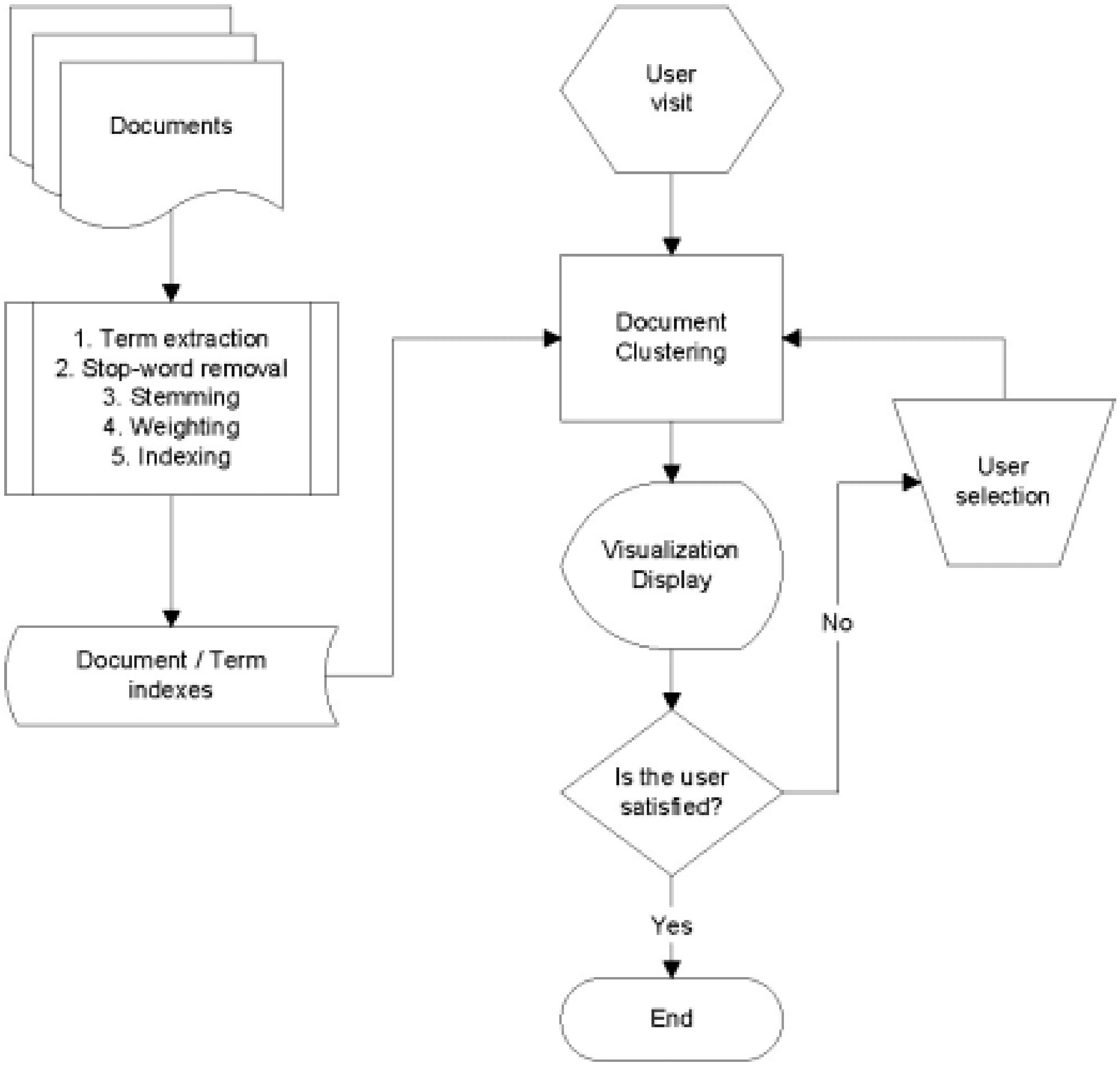
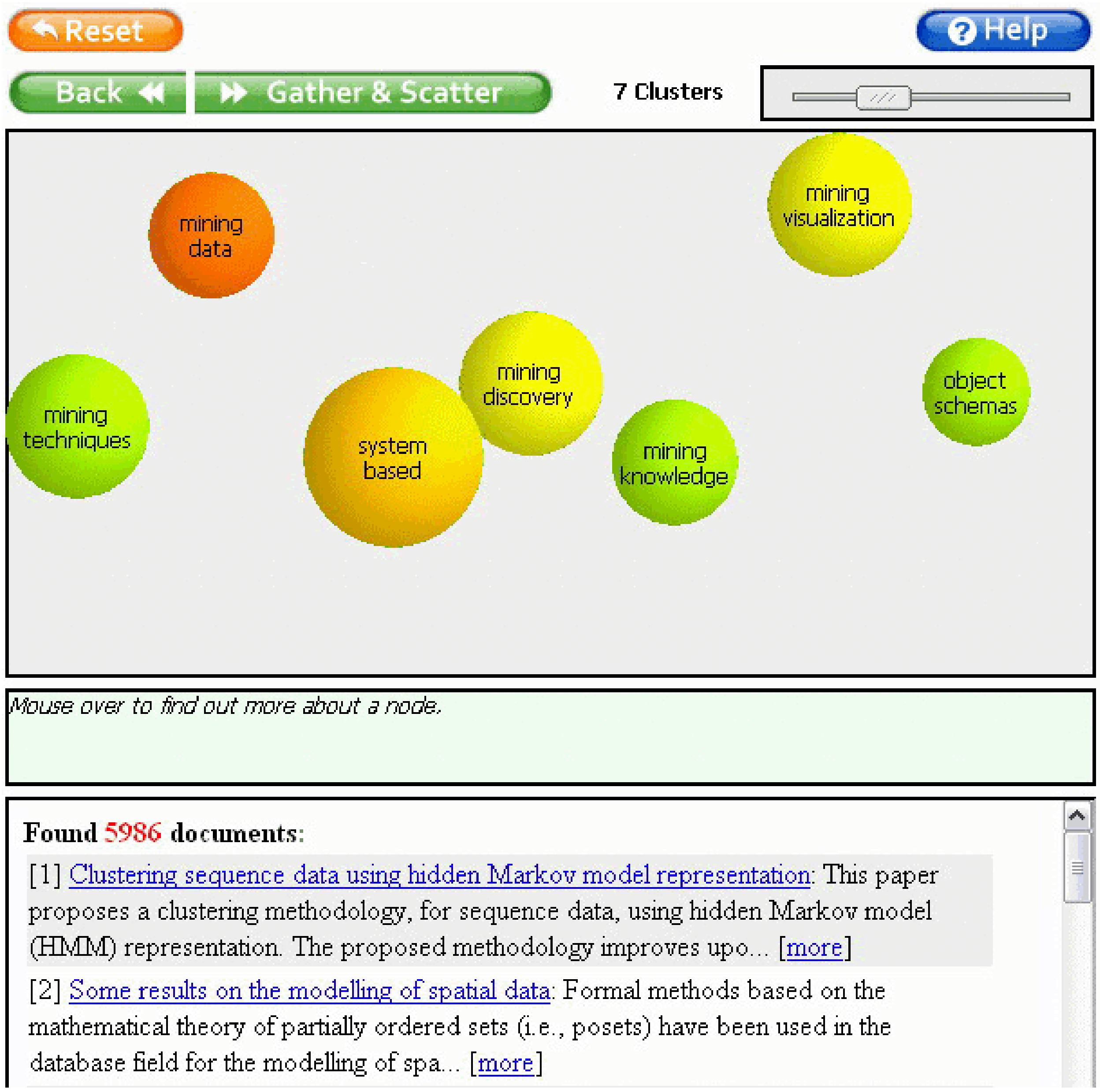


Figure 4: System information flow

System User Interface

Figure 5 shows the prototype interface, which is purely HTML-based and reside in a Web browser. The Scatter/Gather browser has successfully integrated two text corpora respectively: one is a computer/information science literature of 6,000 text documents manually collected; the other is a modified version of the TREC 2005 High Accuracy Retrieval from Documents (HARD) track containing 33,660 news articles. The two versions share the same interface and are available on the public Web through: http://lair.ils.unc.edu/sgbrowser/browser/. As shown in Figure 5, the user interface includes the following elements.



Found 5986 documents:

[1] Clustering sequence data using hidden Markov model representation: This paper proposes a clustering methodology, for sequence data, using hidden Markov model (HMM) representation. The proposed methodology improves upo... [more]

[2] Some results on the modelling of spatial data: Formal methods based on the mathematical theory of partially ordered sets (i.e., posets) have been used in the database field for the modelling of spa... [more]

Figure 5: System User Interface

- Back button: can be used if the user wants to return to the previous cluster selection at any time in the Scatter/Gather process.
- Cluster: visualized by color-coded circles, is a group of documents in the database which are similar/related to each other.
- Cluster size: determined by the number of documents that belong to the cluster. Note that it is based on a log function of the number in order to visualize very large and very small clusters.
- Cluster color: determined based on the homogeneity of the given cluster–the warmer the color, the higher the level of homogeneity of that cluster.

- Cluster position: determined by the similarity of two given clusters–the closer the clusters, the higher their similarity.
- Gather & Scatter button: function for iterating through the database after selecting the desired cluster/s. If the button is clicked without selecting any of the clusters, an error message is displayed asking the user to select at least one cluster.
- Home button: takes the user back to the main introduction page of the Scatter/Gather browser.
- Reset button: resets the Scatter/Gather browser to its default initial state.
- Slider: used to increase or decrease the number of clusters to be displayed on the screen after each iteration. Moving the slider to the left decreases the number of clusters desired and vice-versa.

Using the above mentioned functionalities, the system operates in the following way. The initial index page of the Scatter/Gather browser shows, by default, seven clusters/nodes displaying seven main topics of the text collection. These clusters are arranged near or away from each other, based on the similarity of the associated documents. Moving the cursor over a specific cluster displays more information about it in the middle window.

The list of articles related to the shown clusters is displayed in the bottom window. The initial article list shows the ten most relevant documents with brief descriptions and links to related, detailed information. Links to additional document lists appear at the bottom of this current list.

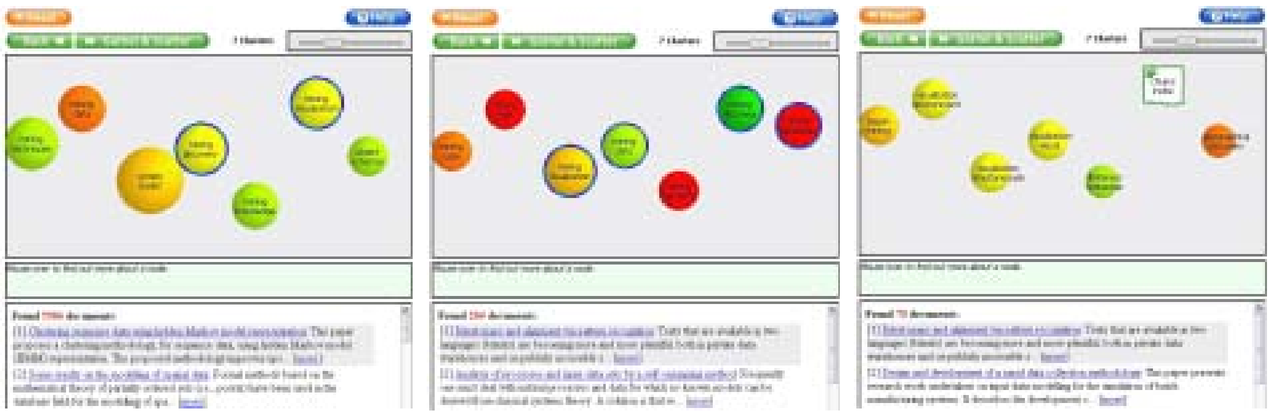

(a) Session 1 (b) Session 2 (c) Session 3

Figure 6: Scatter/Gather Iterations

Figure 6 exemplifies Scatter/Gather search iterations. For searching on a desired topic, the user selects one or more clusters by clicking on the cluster/s. A blue border appears around the selected cluster/s, identifying it as chosen for further iteration. To deselect the cluster, the user clicks again on the same cluster and the blue border disappears. To produce the iteration, the user presses the Gather & Scatter button, located on the top left side of the window. This produces a new display of clusters, showing information related to the selected cluster/s. The number of clusters to be displayed can be changed (from 3 to 15) at any point in the process. This is done by means of the slider provided on the top right side of the window.

Conclusion

We have designed a new algorithm for Scatter/Gather browsing, namely, the LAIR2 clustering, which achieves near constant response time in each Scatter/Gather iteration. This algorithm requires the construction of a cluster hierarchy using any hierarchical clustering algorithms in the offline phase. A linear table is then used to store the cluster hierarchy for the online phase clustering.

On a data collection containing tens of thousands documents, the fast online clustering algorithm runs several hundred times faster than the parallel Buckshot algorithm. When the size of data collection increases to hundreds of thousands, the clustering time of the LAIR2 algorithm is still satisfactory. This efficiency promises real world applications that deliver responsive services to the end users. We have implemented an online visualization prototype, namely, the LAIR2 Scatter/Gather browser, to demonstrate its utility. Future research will involve evaluation of LAIR2 clustering quality and usability of the Scatter/Gather visualization.

Acknowledgments

Authors acknowledge NSF grant #0333623 for support. We thank Alex Berry and Sujit Gadkari for their collaboration on the Scatter/Gather browser implementation. We also appreciate valuable discussions with Kiduk Yang, Diane Kelly, and Cassidy Sugimoto.

References

Can, F. (1993). Incremental clustering for dynamic information processing. *ACM Transaction on Information Systems*, *11*(2), 143–164.

Crouch, D. B., Crouch, C. J., & Andreas, G. (1989). The use of cluster hierarchies in hypertext information retrieval. In *HYPERTEXT '89: Proceedings of the second annual ACM conference on Hypertext* (pp. 225–237). New York, NY, USA: ACM Press.

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

Cutting, D. R., Karger, D. R., & Pedersen, J. O. (1993). Constant interaction-time Scatter/Gather browsing of very large document collections. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 126–134). New York, NY, USA: ACM Press.

Hearst, M. A., & Pedersen, J. O. (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* (pp. 76–84). New York, NY, USA: ACM Press.

Jensen, E. C., Beitzel, S. M., Pilotto, A. J., Goharian, N., & Frieder, O. (2002). Parallelizing the Buckshot algorithm for efficient document clustering. In *CIKM '02: Proceedings of the eleventh international conference on information and knowledge management* (pp. 684–686). New York, NY, USA: ACM Press.

Lipscomb, C. E. (2000, Jul). Medical subject headings (mesh). *Bull Med Libr Assoc*, *88*(3), 265–266.

Liu, Y., Mostafa, J., & Ke, W. (2007, November). *A fast online clustering algorithm for Scatter/Gather browsing* (Tech. Rep. No. TR-2007-06). Chapel Hill, NC, U.S.A.: UNC School of Information and Library Science.

Sahoo, N., Callan, J., Krishnan, R., Duncan, G., & Padman, R. (2006). Incremental hierarchical clustering of text documents. In *Cikm '06: Proceedings of the 15th acm international conference on information and knowledge management* (pp. 357–366). New York, NY, USA: ACM Press.