Generating, Visualizing, and Evaluating High-Quality Clusters for Information Organization

Javed Aslam, Katya Pelekhov, and Daniela Rus

Department of Computer Science, Dartmouth College, Hanover NH 03755
{jaa,katya,rus}@cs.dartmouth.edu

Abstract. We present and analyze the star clustering algorithm. We discuss an implementation of this algorithm that supports browsing and document retrieval through information organization. We define three parameters for evaluating a clustering algorithm to measure the topic separation and topic aggregation achieved by the algorithm. In the absence of benchmarks, we present a method for randomly generating clustering data. Data from our user study shows evidence that the star algorithm is effective for organizing information.

1 Introduction

Modern information systems have vast amounts of unorganized data. Users often don't know what they need until they need it. In dynamic, time-pressured situations such as emergency relief for weather disasters, presenting the results of a query as a ranked list of hundreds of titles is ineffective. To cull the critical information out of a large set of potentially useful sources we need methods for organizing as accurately as possible the data and ways of visualizing this organization flexibly.

We present a paradigm for organizing data that can be used as a preprocessing step in a static information system or as a post-processing step on the specific documents retrieved by a query. As a pre-processor, this system assists users with deciding how to browse the corpus by highlighting relevant topics and irrelevant subtopics. Such clustered data is useful for narrowing down the corpus over which detailed queries can be formulated. As a post-processor, this system classifies the retrieved data into clusters that capture topic categories and subcategories.

Our clustering method is called the star algorithm. The star algorithm gives a hierarchical organization of a collection into clusters. Each level in the hierarchy is determined by a threshold for the minimum similarity between pairs of documents within a cluster at that particular level in the hierarchy. This method conveys the topic-subtopic structure of the corpus according to the similarity measure used. Our implementation uses a modification of the Smart [Sal91] system and the underlying cosine metric. The star algorithm is accurate in that it produces dense clusters that approximate cliques with provable guarantees on the pairwise similarity between cluster documents, yet are computable in $O(N^2)$, where N is the number of documents. The documents in each cluster are tightly inter-related and a minimum similarity distance between all the document pairs in the cluster is guaranteed. This resulting structure reflects the underlying topic structure of the data. A topic summary for each cluster is provided by the center of the underlying star for the cluster.

To examine the performance of the star information organization system we developed a visualization method for data organized in clusters that presents users with three views of the data: a list of text titles; a Euclidean projection of the clusters in the plane as disks (of radius proportional to the size of the cluster) that are separated by distances proportional to the similarity distance between the clusters, and a graph that shows the similarity relationships between the documents. The user can examine each view and select individual objects in the view. For instance, the user may select the largest disk in the projection window. This causes the titles of the documents and their corresponding vertices to be highlighted in the title and graph views. The user may adjust interactively the thresholding parameter for clustering.

To evaluate the performance of this organization system we defined a precision-recall measure for clustering. We also identified that the intersection point between the *precision curve* and the *recall curve* is the *critical point* for measuring the overall performance for information organization tasks. In the absence of benchmarks for clustering we developed two methods for randomly generating benchmarks. We measured the precision-recall of our algorithm against this data and found evidence that our algorithm has a high expected critical point. Depending on how much noise there is in the data, this value is at least 0.8. To validate these results, we did a user study on a collection of technical reports. We compared the user clusters against the system clusters and found further evidence that the star algorithm has good performance.

The experimental data we gathered and our user studies give strong positive evidence that clustering is a useful method for applications that require organizing data according to topic. Such applications typically require the algorithm to have high recall, as in the case of browsing and data reduction. Hearst and Pedersen [HP96] have already provided evidence that the clustering mechanism of Scatter/Gather is useful for high-recall tasks. Scatter/Gather uses fractionation to compute nearest-neighbour clusters. It is expected to produce clusters with loosely connected documents. Our clustering method trades-off performance for accuracy and yields tightly connected clusters. This, along with our preliminary experimental studies, encourages us to think that clustering algorithms with guarantees on the accuracy of the clusters will support the cluster hypothesis and thus assist in tasks that require high precision.

This paper is organized as follows. We first review related work. We then introduce our clustering algorithms. We continue by describing our implementation and visualization. Finally, we explain our performance measures and discuss experimental data.

2 Previous Work

There has been extensive research on clustering and applications to many domains. For a good overview see [JD88]. For a good overview of using clustering in information retrieval see [Wil88].

The use of clustering in information retrieval was mostly driven by the cluster hypothesis [Rij79] which states that relevant documents tend to be more closely related to each other than to non-relevant documents. Efforts have been made to find whether the cluster hypothesis is valid. Voorhees [Voo85] discusses a way of evaluating whether the cluster hypothesis holds and shows negative results. Croft [Cro80] describes a method for bottom-up cluster search that could be shown to outperform a full ranking system for the Cranfield collection. Willett's study [Wil88] shows that the methods he surveys do not outperform non-clustered search methods. In [JR71] Jardine and van Rijsbergen show some evidence that search results could be improved by clustering. Hearst and Pedersen [HP96] re-examine the cluster hypothesis and conclude that it holds for tasks that require high recall, such as browsing.

Our work on clustering presented in this paper provides further evidence that clustering is good for applications where the recall is important. We also show that by trading off some of the performance of a fast system such as Scatter/Gather[1] [CKP93] with computation to ensure cluster accuracy, (that is, to guarantee a minimum similarity between all pairs of documents in a cluster) clusters can also be good for tasks where precision is important. To compute accurate clusters, we formalize clustering as covering graphs by cliques. Covering by cliques is NP-complete, and thus intractable for large document collections. Recent graph-theoretic results have shown that the problem can't even be approximated in polynomial time [LY94,Zuc93]. Recent results for covering graphs by dense subgraphs [KP93] are encouraging. We used a cover by dense subgraphs that are star-shaped[2]. We show that this algorithm is an accurate and efficient approximation of cliques, propose a measure for the quality of the clusters, and provide experimental data.

3 Clustering Applications

The main application we have in mind for clustering is in information organization. Information organization can be used for browsing. If the clusters capture the topic structure of the collection, organization can also be used to narrow the search domain of a query and to organize the results retrieved in response to a query. We also believe that tightly connected clusters (unlike loosely connected clusters such as those obtained by a nearest-neighbour method or a single link method) can also be used to improve retrieval as the cluster hypothesis suggests, by returning the clusters corresponding to the top-most ranked documents. For

[1] Scatter/Gather uses fractionation to compute nearest-neighbor clusters.

[2] In [SJJ70] stars were also identified to be potentially useful for clustering.

our star-algorithm, an alternative is to return an entire cluster only when a top-ranked document is the center of a star. We are currently collecting data for this application.

4 Our Clustering Method

In this section we motivate and present two algorithms for organizing information systems. The first of our algorithms is very simple and efficient, and our second algorithm, while somewhat slower, has the advantage of being more accurate.

We formulate our problem by representing an information system by its *similarity graph*. A similarity graph is an undirected, weighted graph $G = (V, E, w)$ where vertices in the graph correspond to documents and each weighted edge in the graph corresponds to a measure of similarity between two documents. We measure the similarity between two documents by using the cosine metric in the vector space model of the Smart information retrieval system [Sal91,SM83]. G is a complete graph with edges of varying weight. An organization of the graph that produces reliable clusters of similarity σ (*i.e.*, clusters where documents pairwise have similarities of at least σ) can be obtained by performing a *minimum clique cover* of all edges whose weights are above the threshold σ. The following algorithm can be used to produce a hierarchy of such organizations which we call *summaries*:

For any threshold σ:

1. Let $G_\sigma = (V, E_\sigma)$ where $E_\sigma = \{e : w(e) \geq \sigma\}$.
2. Compute the minimum clique cover of G_σ.
3. Represent each clique by a sequence of representative terms or by any document in the clique.

Fig. 1. The clique-cover algorithm

This algorithm has three nice features. First, by using cliques to cover the similarity graph, we are guaranteed that all the documents in a cluster have the desired degree of similarity. Second, covering the edges of the graph allows vertices to belong to *several* clusters. Documents can be members of multiple clusters, which is a desirable feature when documents have multiple subthemes. Third, this algorithm can be iterated for a range of thresholds, effectively producing a hierarchical organization structure for the information system. Each level in the hierarchy summarizes the collection at a granularity provided by the threshold.

Unfortunately, this approach is computationally intractable. For real corpora, these graphs can be very large. The clique cover problem is NP-complete, and it does not admit polynomial-time approximation algorithms [LY94,Zuc93].

While we cannot perform a clique cover nor even approximate such a cover, we can instead cover our graph by *dense subgraphs*. What we lose in intra-cluster similarity guarantees, we gain in computational efficiency. In the sections that follow, we describe two such covering algorithms and analyze their performance and efficiency.

4.1 Covering with Star-Shaped Subgraphs

While covering the thresholded similarity graph with cliques has many desirable properties as described in the previous section, finding such a covering is, unfortunately, computationally intractable. We shall instead find a clustering of a set of documents by covering the associated thresholded similarity graph with *star-shaped subgraphs*. A star-shaped subgraph on $m+1$ vertices consists of a single *star center* and m *satellite vertices*, where there exist edges between the star center and each of the satellite vertices. While finding cliques in the thresholded similarity graph G_σ *guarantees* a pairwise similarity between documents of at least σ, it would appear at first glance that finding star-shaped subgraphs in G_σ would provide similarity guarantees between the star center and each of the satellite vertices, but no such similarity guarantees *between satellite vertices*. However, by investigating the geometry of our problem in the vector space model, we can derive a *lower bound* on the similarity between satellite vertices as well as provide a formula for the *expected* similarity between satellite vertices. The latter formula predicts that the pairwise similarity between satellite vertices in a star-shaped subgraph is high, and together with empirical evidence supporting this formula, we shall safely conclude that covering G_σ with star-shaped subgraphs is a reliable method for clustering a set of documents.

Consider three documents C, S_1 and S_2 which are vertices in a star-shaped subgraph of G_σ, where S_1 and S_2 are satellite vertices and C is the star center. By the definition of a star-shaped subgraph of G_σ, we must have that the similarity between C and S_1 is at least σ and that the similarity between C and S_2 is also at least σ. In the vector space model, these similarities are obtained by taking the cosine of the angle between the vectors associated with each document. Let α_1 be the angle between C and S_1, and let α_2 be the angle between C and S_2. We then have that $\cos\alpha_1 \geq \sigma$ and $\cos\alpha_2 \geq \sigma$. Note that the angle between S_1 and S_2 can be at most $\alpha_1 + \alpha_2$, and therefore the similarity between S_1 and S_2 must be at least

$$\cos(\alpha_1 + \alpha_2) = \cos\alpha_1 \cos\alpha_2 - \sin\alpha_1 \sin\alpha_2.$$

Thus, we have a provable lower bound on the similarity between satellite vertices in a star-shaped subgraph of G_σ. If $\sigma = 0.7$, $\cos\alpha_1 = 0.75$ and $\cos\alpha_2 = 0.85$, for instance, we can conclude that the similarity between the two satellite vertices must be at least[3]

$$(0.75) \cdot (0.85) - \sqrt{1-(0.75)^2}\sqrt{1-(0.85)^2} \approx 0.29.$$

[3] Note that $\sin\theta = \sqrt{1-\cos^2\theta}$.

Note that while this may not seem very encouraging, the above analysis is based on absolute worst-case assumptions, and in practice, the similarities between satellite vertices are much higher. We further undertook a study to determine the *expected* similarity between two satellite vertices. By making the mathematical assumption that "similar" documents are essentially "random perturbations" of one another in the vector space model, we were able to derive the following formula for the *expected* similarity between two satellite vertices:

$$\cos\alpha_1 \cos\alpha_2 + \frac{\sigma}{1+\sigma} \sin\alpha_1 \sin\alpha_2 .$$

Note that for the previous example, the above formula would predict a similarity between satellite vertices of approximately 0.78. We have tested this formula against real data, and the results of the test with the MEDLINE data set are shown in Figure 2. In this plot, the x- and y-axes are similarities between cluster centers and satellite vertices, and the z-axis is the actual mean squared prediction error of the above formula for the similarity between satellite vertices. Note that the absolute error (roughly the square root of the mean squared error) is quite small (approximately 0.13 in the worst case), and for reasonably high similarities, the error is negligible. From our tests with real data, we have concluded that this formula is quite accurate, and hence we can conclude that star-shaped subgraphs are reasonably "dense" in the sense that they imply relatively high pairwise similarities between documents.

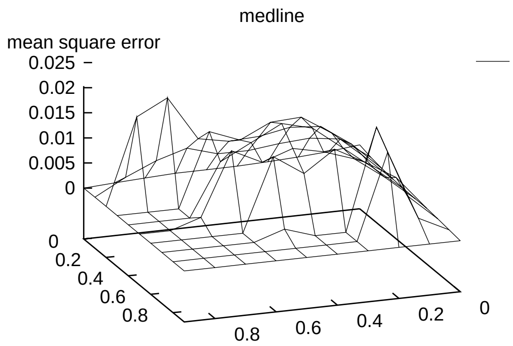

Fig. 2. This figure shows the error for a 6000 abstract subset of MEDLINE.

4.2 The Star and Star+ Algorithm

Motivated by the discussion of the previous section, we now present the *star algorithm* which can be used to organize documents in an information system.

The star algorithm is based on a greedy cover of the thresholded similarity graph by star-shaped subgraphs; the algorithm itself is summarized in Figure 3 below.

For any threshold σ:

1. Let $G_\sigma = (V, E_\sigma)$ where $E_\sigma = \{e : w(e) \geq \sigma\}$.
2. Let each vertex in G_σ initially be *unmarked*.
3. Calculate the degree of each vertex $v \in V$.
4. Let the vertex of highest degree be a star center, and construct a cluster from the star center and its associated satellite vertices. Mark each node in the newly constructed cluster.
5. Set the star center's degree to zero and decrement each satellite vertex's degree by one.
6. Repeat steps 4 and 5 until all nodes are marked.
7. Represent each cluster by the document corresponding to its associated star center.

Fig. 3. The star algorithm

Implemented properly, the star algorithm is very efficient—it can be made to run in time *linear* in the number of edges of the graph, which is $O(N^2)$ where N is the number of vertices in the graph. We will save our discussion of the performance of the star algorithm for the following sections, but as motivation for the subsequent improved algorithm, we will note now that the star algorithm as described above performed very well in a small user study, though somewhat less well on randomly generated data. To improve the performance of the star algorithm, we must improve the "quality" of the clusters it generates. We can improve the quality of the clusters generated by being somewhat more selective about the vertices included in a newly generated cluster. In the augmented *star+ algorithm* described below, a satellite vertex is only included in a cluster if at least one-third of the other candidate satellite vertices have a similarity of at least σ with respect to the satellite vertex in question. (In this heuristic, the parameter "one-third" was arrived at empirically.)

While the star+ algorithm is somewhat slower than the original star algorithm, it produces more accurate clusters, slightly outperforming the star algorithm in a user study and markedly outperforming the star algorithm on randomly generated data. In the sections that follow, we describe our performance analysis of these algorithms in detail.

5 System Description

We have implemented a system for organizing information that uses the star and star+ algorithms. This organization system (that is the basis for the experiments described in this paper) consists of an augmented version of the Smart

For any threshold σ:

1. Let $G_\sigma = (V, E_\sigma)$ where $E_\sigma = \{e : w(e) \geq \sigma\}$.
2. Let each vertex in G_σ initially be *unmarked*.
3. Calculate the degree of each vertex $v \in V$.
4. Let the vertex of highest degree be a star center, and include this vertex in a newly constructed cluster. For each satellite vertex, include the vertex in the cluster if there exist edges incident to this vertex from at least 1/3 of the other satellite vertices. Mark each node in the newly constructed cluster.
5. Set the star center's degree to zero and decrement the degree of each other vertex in the cluster by one.
6. Repeat steps 4 and 5 until all nodes are marked.
7. Represent each cluster by the document corresponding to its associated star center.

Fig. 4. The star+ algorithm

system [Sal91,All95], a user interface we have designed, and an implementation of the star and star+ algorithms on top of Smart. To index the documents we used Smart search engine with a cosine normalization weighting scheme. We enhanced Smart to compute a document to document similarity matrix for a set of retrieved documents or a whole collection. The similarity matrix is used to compute clusters and to visualize the clusters. The user interface is implemented in Tcl/Tk.

The organization system can be run on a whole collection, on a specified subcollection, or on the collection of documents retrieved in response to a user query. Users can input queries by typing in free text. They have the choice of specifying several corpora. This system supports distributed information retrieval but in this paper we do not focus on distribution and we assume only one centrally located corpus. In response to a user query, Smart is invoked to produce a ranked list of the top most relevant documents, their titles, locations and document-to-document similarity information. The similarity information for the entire collection, or for the collection computed by the query engine is provided as input to the star (or star+) algorithm. This algorithm returns a list of clusters and marks their centers.

5.1 Visualization

We developed a visualization method for organized data that presents users with three views of the data (see Figure 5): a list of text titles, a graph that shows the similarity relationship between the documents, and a graph that shows the similarity relationship between the clusters. These views provide users with summaries of the data at different levels of detail: text, document, and topic and facilitate browsing by topic structure.

The connected graph view (inspired by [All95]) has nodes corresponding to the retrieved documents. The nodes are placed in a circle, with nodes corresponding to the same cluster placed together. Gaps between the nodes allow us to identify clusters easily. Edges between nodes are color coded according to the similarity between the documents. Two slider bars allow the user to establish minimal and maximal weight of edges to be shown.

Another view presents clusters as disks of a size proportional to the size of the corresponding cluster. The distance between two clusters is defined as a distance between the central documents and captures the topic separation between the clusters. Simulated annealing is used to find a cluster placement that minimizes the sum of relative distance errors between clusters. We selected a cooling schedule $\alpha(t) = t/(1 + \beta t)$, where $\beta = 10^{-3}$, initial temperature is 500 and the freezing point is 10^{-2}. This setting provides a good placement when the number of clusters returned by the algorithm is small. This algorithm is fast and its running time does not depend on the number of clusters. When the number of clusters is large, the ellipsoid-based method for Euclidean graph embeddings described in [LLR95] can be used instead.

All three views and a title window allow the user to select an individual document or a cluster. Selection made in one window is simultaneously reflected in the others.

6 Evaluation

Our hypothesis for measuring the performance of a clustering algorithm is that (1) all the different topics should be separated in different clusters, and (2) all the documents relevant to a topic should be aggregated together. We call (1) the *separation* property and (2) the *aggregation* property. The main goal of our experiments is to find whether the star algorithm has good separation and aggregation. A clustering algorithm that guarantees both aggregation and separation is well-suited to improve recall-oriented tasks as well as precision-oriented tasks.

We define three measures, *Precision*, *Recall*, and *critical point* for evaluating the separation and aggregation of our clustering method by drawing inspiration from the precision-recall measures for information retrieval.

Our measures are defined in terms of a "correct" clustering. In the absence of any benchmarks for clustering, we tried to produce one on the MEDLINE collection by using the humanly-assigned indices. We found that if we use the human indices only as a basis for clustering, the resulting clusters do not make sense. This limited our evaluation to relatively small collections (162 documents) that humans could index to produce "correct" clusters, and to randomly generated clustered data. This data in described later in this section.

6.1 Precision-Recall Measures

Precision and Recall for clustering are defined relative to a *correct* clustering of the same data. Let $C_{correct}$ denote the correct clustering of the data and $C_{computed}$ denote the computed clustering. For each document d in the collection we can find the set of clusters $S_{correct} \subset C_{correct}$ and $S_{computed} \subset C_{computed}$ that contain the document. The precision P_d and recall R_d for this document are computed as:

$$P_d = \frac{S_{correct} \cap S_{computed}}{S_{computed}}.$$

$$R_d = \frac{S_{correct} \cap S_{computed}}{S_{correct}}.$$

The precision (respectively, recall) of the clustering algorithm is then computed as the average of the precision (respectively, recall) values for all documents in the corpus:

$$Precision = \frac{\sum_{i=1}^{n} P_{d_i}}{n}.$$

$$Recall = \frac{\sum_{i=1}^{n} R_{d_i}}{n}.$$

Different thresholds for the minimum similarity between two documents in the corpus result in different precision and recall values. If we plot precision and recall against the threshold value we obtain precision and recall curves.

6.2 The Critical Point

Under these measures, a trivial algorithm that clusters each document by itself has high precision but poor recall. A trivial algorithm that clusters all the documents in one cluster has high recall but poor precision. It is easy to produce clustering algorithms that achieve high performance on the recall curve or on the precision curve, but not both. High precision guarantees that different topics are separated in different clusters. High recall guarantees that all the documents relevant to a topic are grouped in the same cluster. We would like to have both good separation between topics and guarantees that all the documents relevant to a topic are aggregated together. We propose a third parameter called the *critical point* as a measure of this idea. The critical point is defined as the intersection point of the precision curve and the recall curve. High critical points guarantee both topic separation and topic aggregation.

6.3 Data Generation

In the absence of any suitable benchmarks by which to test our clustering algorithms, we chose to test our algorithms using data that we either generated or collected ourselves. The data that we used has two forms: first, we generated clustering data randomly in two different ways, and second, we performed a small user study with a real document collection. The former allowed us to have complete control over an arbitrarily large corpus, while the latter allowed us to test against user expectations, albeit for small collections. In the sections that follow, we explain and give the results of our studies with randomly generated data and a real collection of documents.

Generating Random Data Our first experiments involved testing our algorithms against randomly generated clustering data. We generated clustering data by essentially *constructing the similarity graph* for a hypothetical document collection. Our data generation algorithm admitted the following parameters: collection size, mean cluster size and variance, mean cluster overlap and variance, mean intra-cluster edge weight and variance, and mean inter-cluster edge weight and variance. Our procedure for randomly constructing a similarity graph can be divided into two phases: in the first phase, the overlapping clusters of vertices are generated, and in the second phase, appropriate edge weights are generated. To generate the clusters of vertices, a sequence of *cluster size* and *cluster overlap* numbers are generated according to the parameters specified to the algorithm. For example, the following are typical cluster size and cluster overlap sequences:

Cluster size: $24, 12, 22, 29, 16, \ldots$
Cluster overlap: $3, 5, 0, 6, 4, \ldots$

From this data, the proposed clusters can be generated. Cluster 1 would consist of vertices 1 through 24, cluster 2 would consist of vertices 22 through 33, cluster 3 would consist of vertices 29 through 50, and so on—the size of each successive cluster would be dictated by the sequence of cluster sizes, and the overlap between consecutive clusters would be dictated by the sequence of cluster overlaps.

Following the generation of the clusters themselves, all of the edge weights are then constructed. For each pair of vertices, a random edge weight is generated according to either the intra- or inter-cluster distribution, respectively, depending on whether the pair of vertices belong to the same or different clusters. Having generated a similarity graph with known clusters, we can test various algorithms against the known clustering and measure performance according the to the precisions-recall metrics defined above. Note that by carefully setting the mean and variance of the intra- and inter-cluster distributions, one can create similarity graphs with a specified fraction of *faulty data*. If the intra- and inter-cluster distributions *overlap*, then a fraction of the intra-cluster edge weights will look "more like" inter-cluster edge weights, and vice versa. Such a scheme allows one to simulate real, faulty data, and our studies with such randomly generated data are described below.

Experimental Results on Random Data We generated two data sets according to the algorithm described above by varying the percentage of faulty data. The first set has 15% faulty data (that is, overlap between the inter-cluster edges and intr-cluster edges) and the second set has 20% faulty data. We used these clusters as the correct clusters in our precision-recall measures and evaluated the performance of the star and star+ algorithm on this data. The precision-recall curves are shown in Figure 6.

We note that the critical point for the star algorithm is medium at 0.5 for the first data set and 0.38 for the second (more faulty) data set. The critical point of for the star+ algorithm is at 0.9 for the first data set and 0.8 for the second set. We are very encouraged by these high values.

Generating Random Data on the Sphere While the random data generation procedure described above is very useful in evaluating clustering algorithms, the data created will not necessarily meet the geometric constraints imposed by the vector space model on real data. In this section, we briefly describe a procedure for generating random clustering data which does meet the geometric constraints imposed by the vector space model.

In the vector space model, documents are represented by vectors in a high-dimensional space, and the similarity between pairs of documents is given by the cosine of the angle between the associated vectors. In the previous sections, we described a mechanism for generating the *similarity graph* associated with a collection. In this new data generation procedure, we instead randomly create the *vectors* in high-dimensional space which correspond to documents, and then construct the associated similarity graph from these vectors. In brief, well-spaced *cluster centers* are generated on a unit sphere of high-dimension, and the clusters of documents themselves are generated by randomly perturbing these cluster centers. By carefully varying the "spacing" of the cluster centers as well as the amount of perturbation allowed in generated the cluster documents, we can again allow for a specified overlap of clusters as well as a varying degree of faulty data. Our experiments with this type of randomly generated clustering data are presented below.

Experimental Results on Random Data on the Sphere We generated two data sets according to the algorithm described above by varying the percentage of faulty data. The first set has 7% faulty data (that is, overlap between the inter-cluster edges and intr-cluster edges) and the second set has 12% faulty data. We used these clusters as the correct clusters in our precision-recall measures and evaluated the performance of the star and star+ algorithm on this data. The precision-recall curves are shown in Figure 7.

We note that the critical point for the star algorithm is medium at 0.55 for the first data set and 0.47 for the second (more faulty) data set. The critical point of for the star+ algorithm is at 0.98 for the first data set and 0.8 for the second set. We are very encouraged by these high values. We generated 12 other data sets by varying the probabilities, the distance between the cluster centers, the minimum similarity within a cluster, the number of clusters, and the number of documents per cluster. The locations of the critical points are shown in Fugure 8. The percentage of faulty data seems to be the most sensitive parameter in these experiments.

6.4 A User Study on Technical Reports

We designed an experiment to compute clusters that are "correct" from the perspective of humans, and used these clusters as the correct clusters in the precision-recall computation.

Our study consisted of four graduate students. These students were presented with 162 abstracts from the computer science technical report collection and were only told to cluster the data. No further instructions on how to do the clusters were given. One of the users missed some of the documents so we discarded his data. We then compared the user clusters among themselves and against the star-clustering. The data from this study is shown in Figure 9. The user data is separated in two groups: (a) two users decided to allow one-document clusters and (b) one user decided to try to cluster all the documents in large clusters. We found that the star algorithm has a high critical point (at 0.6) when compared to the clusters generated by group (a) and a medium critical point (at 0.43) when compared with the clusters generated by group (b). This suggests that the star algorithm has good separation and aggregation of data and is thus well-suited for information organization.

7 Discussion

We have presented and analyzed a clustering algorithm. We have discussed methods for evaluating clustering and for generating benchmarks for clustering. Our user studies present positive evidence that the star clustering algorithm can be used to organize information and further support the cluster hypothesis. Our work extends previous results [HP96] that support using clustering for browsing applications. We argue that by using a clustering algorithm that guarantees the cluster quality through high separation and aggregation, clustering is also beneficial for applications that require high precision.

In the future we hope to do more detailed user studies. In the absence of benchmarks this is a tedious task, as reading and manually organizing thousands of documents is time consuming. We also plan to develop experiments that will address directly the benefits of clustering for retrieval, browsing, and data reduction tasks. Another domain of great interest to us is developing on-line clustering algorithms that will be the basis for self-organizing dynamic information systems.

Acknowledgements

We thank Jonathan Bredin, Mark Montague, and Brian Premore for doing the user study. This work is supported in part by the Navy under contract ONR N00014-95-1-1204 and Rome Labs contract F30602-98-C-0006.

References

[All95] J. Allan. *Automatic hypertext construction*. PhD thesis. Department of Computer Science, Cornell University, January 1995.

[Cro80] W. B. Croft. A model of cluster searching based on classification. *Information Systems*, 5:189-195, 1980.

[Cro77] W. B. Croft. Clustering large files of documents using the single-link method. *Journal of the American Society for Information Science*, pp189-195, November 1977.

[CKP93] D. Cutting, D. Karger, and J. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of the* 16^{th} *SIGIR*, 1993.

[HP96] M. Hearst and J. Pedersen. Reexamining the cluster hypothesis: Scatter/Gather on Retrieval Results. In *Proceedings of the* 19^{th} *SIGIR*, 1996.

[JD88] A. Jain and R. Dubes. *Algorithms for Clustering Data*, Prentice Hall 1988.

[JR71] N. Jardine and C.J. van Rijsbergen. The use of hierarchical clustering in information retrieval, 7:217-240, 1971.

[KP93] G. Kortsarz and D. Peleg. On choosing a dense subgraph. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science (FOCS)*, 1993.

[LC96] A. Leouski and B. Croft. An evaluation of techniques for clustering search results. Technical report, Department of Computer Science, the University of Massachusetts at Amherst, 1996.

[LLR95] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica* 15(2):215-245, 1995.

[LY94] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM* 41, 960–981, 1994.

[Rij79] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[RA95] D. Rus and J. Allan. Structural queries in electronic corpora. In *Proceedings of DAGS95: Electronic Publishing and the Information Superhighway*, May 1995.

[Sal89] G. Salton. *Automatic Text Processing: the transformation, analysis, and retrieval of information by computer*, Addison-Wesley, 1989.

[Sal91] G. Salton. The Smart document retrieval project. In *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 356-358.

[SM83] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[SA93] G. Salton and J. Allan. Selective text utilization and text traversal. In *Hypertext '93 Proceedings*, pages 131-144, Seattle, Washington, 1993.

[SJJ70] K. Spark Jones and D. Jackson. The use of automatically-obtained keyword classifications for information retrieval. *Inform. Stor. Retr.* 5:174-201, 1970.

[Tur90] H. Turtle. Inference networks for document retrieval. PhD thesis. University of Massachusetts, Amherst, 1990.

[VGJ95] E. Voorhees, N. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In *Proceedings of the* 18^{th} *SIGIR*, Seattle, WA, 1995.

[Voo85] E. Voorhees. The cluster hypothesis revisited. In *Proceedings of the* 8^{th} *SIGIR*, pp 95-104, 1985.

[Wil88] P. Willett. Recent trends in hierarchical document clustering: A critical review. *Information Processing and Management*, 24:(5):577-597, 1988.

[Wor71] S. Worona. Query clustering in a large document space. In Ed. G. Salton, *The SMART Retrieval System*, pp 298-310. Prentice-Hall, 1971.

[Zuc93] D. Zuckerman. NP-complete problems have a version that's hard to approximate. In *Proceedings of the Eight Annual Structure in Complexity Theory Conference*, IEEE Computer Society, 305–312, 1993.

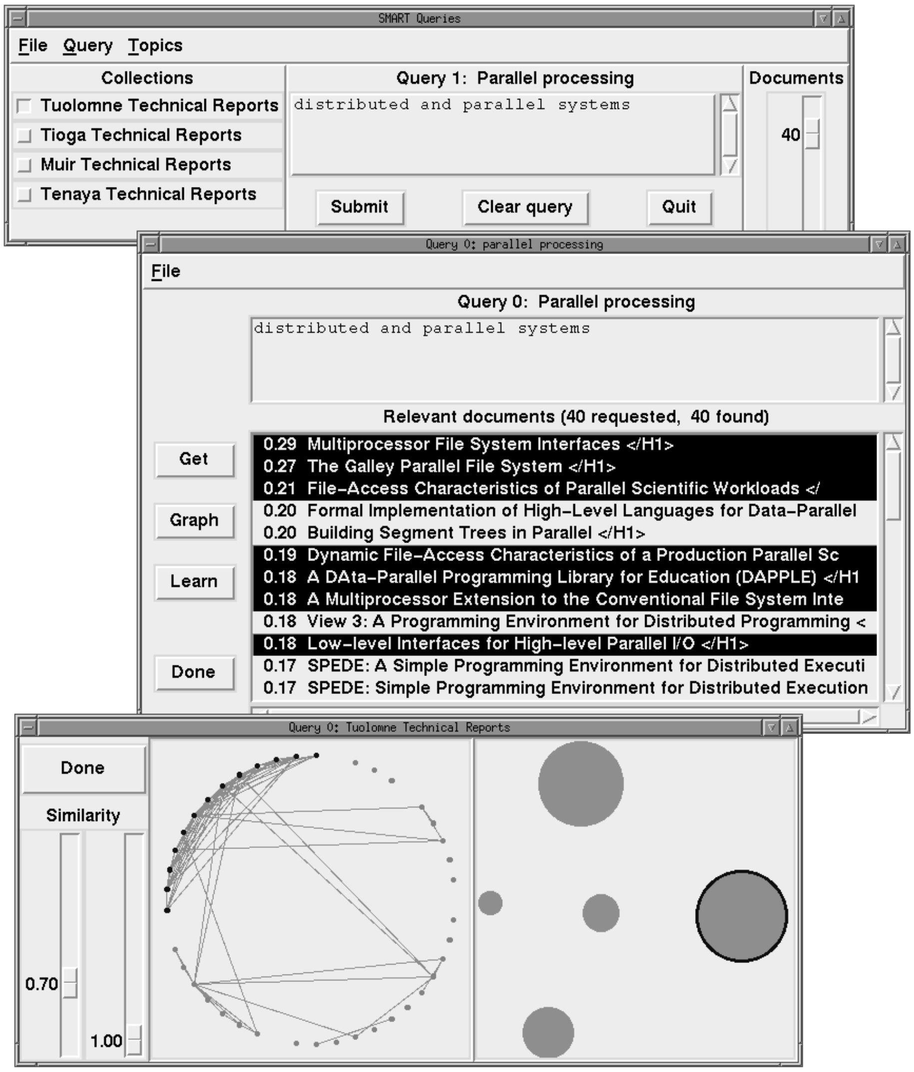

Fig. 5. This is a screen snapshot from a clustering experiment. The top window is the query window. The middle window consists of a ranked list of documents that were retrieved in response to the user query. The user my select "get" to fetch a document or "graph" to request a graphical visualization of the clusters as in the bottom window. The left graph displays all the documents as dots around a circle. Clusters are separated by gaps. The edges denote pairs of documents whose similarity falls between the slider parameters. The right graph displays all the clusters as disks. The radius of a disk is proportional to the size of the cluster. The distance between the disks is proportional to the similarity distance between the clusters.

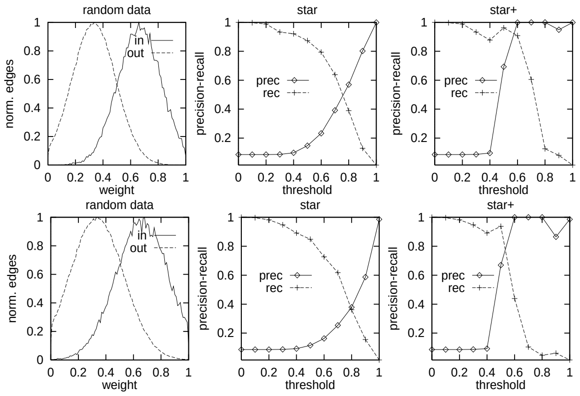

Fig. 6. This figure shows two sets of data. The top set has 15% faulty data and the bottom set has 20% faulty data. For each set we plotted the intra-cluster and inter-cluster edge distribution (left) the precision-recall curves for the star algorithm (middle), and the precision-recall curves for the augmented star algorithm (right).

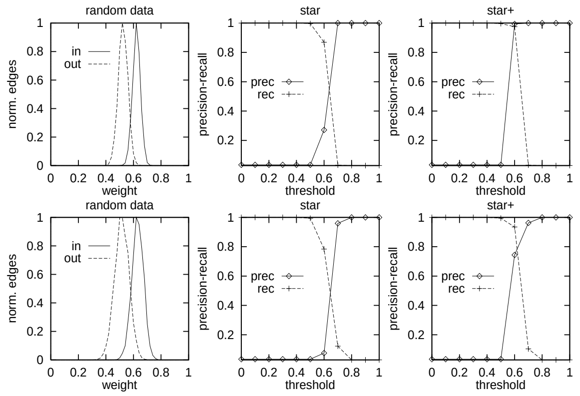

Fig. 7. This figure shows two sets of data. The top set has 7% faulty data and the bottom set has 12% faulty data. For each set we plotted the intra-cluster and inter-cluster edge distribution (left) the precision-recall curves for the star algorithm (middle), and the precision-recall curves for the augmented star algorithm (right).

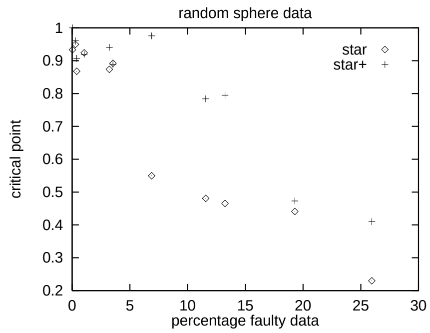

Fig. 8. This figure shows the critical points for 12 experiments with data generated randomly on the sphere.

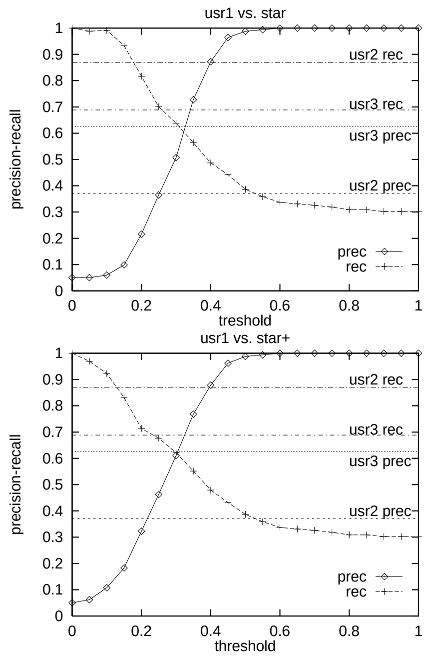

Fig. 9. This figure shows the user-study data for the star and the augmented star algorithm. The plot denoted the precision recall curves achieved by the star+ algorithm against one of user's clusters. The horizontal lines denote the precision and recall values of the other users.