Scalable Information Organization*

Javed Aslam, Fred Reiss, and Daniela Rus
Department of Computer Science
Dartmouth College
Hanover, NH 03755 USA
{*jaa, frr, rus*}*@cs.dartmouth.edu*

Abstract

We present three scalable extensions of the star algorithm for information organization that use sampling. The star algorithm organizes a document collection into clusters that are naturally induced by the topic structure of collection, via a computationally efficient cover by dense subgraphs. We also provide supporting data from extensive experiments.

1 Introduction

Our goal is to develop a completely automated information organization system for digital libraries, automated tools for librarians to classify this information, automatic tools to create reference pointers into such collections, and automated tools that allow users to locate information effectively.

We focus on static and dynamic digital collections of unstructured text. We consider the problem of determining the topic structure of text data, without *a priori* knowledge of the number of topics in the data or any other information about their composition. We assume that the collections may be static (for example, digital legacy collections) or dynamic (for example, news wires). We look to discover hierarchies of topics and subtopics in such text collections. Thus, we develop clustering algorithms that can be used in off-line, on-line, and hierarchical mode. We wish for these algorithms to be fast, scalable, accurate, and to discover the naturally occurring topics in the collection. In our previous work (Aslam et al, 1998; Aslam et al, 1999), we proposed an off-line and an on-line approach based on graph theory. Our algorithms, called the *star clustering* algorithms, compute clusters induced by the *natural* topic structure of the space. Thus, this work is different than previous work in using clustering to organize information (Cutting et al, 1993; Charikar et al, 1997) in that we do not impose the constraint to use a fixed number of clusters. This previous work argues that the star algorithm is simple, efficient, can be used in off-line as well as on-line mode, and it outperforms existing clustering algorithms such as single link, average link, and k-means. In this paper we consider scalability issues in developing an information organization system. We present three different scalable extensions to the star algorithm and show data from extensive experiments.

2 Related Work

There has been extensive research on clustering and applications to many domains (Everitt, 1993; Mirkin 1996; Silverstein and Pedersen 1997; Sibson, 1973; Worona, 1971). For a good overview see (Jain and

*Research partially supported by ONR contract N00014-95-1-1204, Rome Labs contract F30602-98-C-0006, DARPA contract F30602-98-2-0107, Air Force MURI contract F49620-97-1-0382, and NSF grant BCS-9978116.

Dubes, 1988). For a good overview of using clustering in Information Retrieval (IR) see (Willett, 1988). The use of clustering in IR was mostly driven by *the cluster hypothesis* (Rijsbergen, 1979) 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 (Voorhees, 1985) discusses a way of evaluating whether the cluster hypothesis holds and shows negative results. Croft (Croft, 1080) describes a method for bottom-up cluster search that could be shown to outperform a full ranking system for the Cranfield collection. In (Jardine and van Rijsbergen, 1971) Jardine and van Rijsbergen show some evidence that search results could be improved by clustering. Hearst and Pedersen (Hearst and Pedersen, 1996) re-examine the cluster hypothesis by focusing on the Scatter/Gather system (Cutting et al, 1993) and conclude that it holds for browsing tasks.

Systems like Scatter/Gather (Cutting et al, 1993) provide a mechanism for user-driven organization of data in a fixed number of clusters, but the users need to be in the loop and the computed clusters do not have accuracy guarantees. Scatter/Gather uses fractionation to compute nearest-neighbor clusters. Charika et al. (Charikar et al, 1997) consider a dynamic clustering algorithm to partition a collection of text documents into a *fixed number* of clusters. Since in dynamic information systems the number of topics is not known *a priori*, a fixed number of clusters cannot generate a natural partition of the information.

3 Background: The Star Algorithm for Information Organization

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 highest degree unmarked vertex 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. Repeat step 4 until all nodes are marked.
6. Represent each cluster by the document corresponding to its associated star center.

Figure 1: The star algorithm

To compute accurate topic clusters, one possibility is to formalize clustering as covering similarity graphs by cliques. A clique cover will guarantee that its documents are strongly related to each other. Covering by cliques is NP-complete, and thus intractable for large document collections. Unfortunately, it has also been shown that the problem cannot even be approximated in polynomial time (Zuckerman, 1993). We instead propose using a cover by *dense subgraphs* that are *star-shaped* and that can be computed *off-line* for static data and *on-line* for dynamic data. What we lose in intra-cluster similarity guarantees, we gain in computational efficiency.

We represent the document collection as a complete similarity graph, where the vertices correspond to documents and the edges are weighted by a similarity measure. We have used two measures: the cosine metric and an information-theoretic metric.

To compute accurate topic clusters, we create a thresholded similarity graph, where the thresholding parameter is given by the smallest similarity we would like to have between any documents within a topic. We then approximate a clique cover of this graph 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. A greedy algorithm (see Figure 1) computes this cover for static collections. In (Aslam et al, 1998; Aslam et al, 1999) we show an on-line version of this algorithm that supports information organization in dynamic collection.

Star-graph covers are interesting because they provide accuracy guarantees on the computed topics. By investigating the geometry of the problem, we can derive a *lower bound* on the similarity between satellite vertices as well as provide a formula ($\cos\gamma \geq \cos\alpha_1 \cos\alpha_2 + \frac{\sigma}{1+\sigma} \sin\alpha_1 \sin\alpha_2$, where α_1 and α_2 correspond to the similarity between the center and the two satellites and σ is the similarity threshold) for the *expected* similarity between satellite vertices using the cosine metric. This formula predicts that the pairwise similarity between satellite vertices in a star-shaped subgraph is high, and together with empirical evidence supporting this formula (Aslam et al, 1998).

4 Scalable Extensions for the Star Algorithm

For any threshold σ:

1. Let D be a set of n documents sorted in random order in an array.
2. Let s be the sample size.
3. Compute a Star Cover for $D[1..s]$ and let C be the list of star centers of this cover.
4. For each document $D[i]$ in $D[s + 1..n]$
   - For each cluster $C[j]$ in C: if $similarity(D[i], C[j]) > \sigma$ insert $D[i]$ in $C[j]$
   - If $D[i]$ was not inserted in any existing cluster, create a new cluster with $D[i]$ as a center and add this cluster to C.

Figure 2: The sampled star algorithm.

In this section we present three extensions to the star algorithm that optimize its performance. The three algorithms compute approximations to the star cluster but optimize on the size of the similarity matrix used and on the time required to generate it.

Both of the off-line and on-line versions of the star algorithm rely on the existence of the similarity matrix. Similarity matrices can get very large: for a document set with n documents the similarity matrix is $O(n^2)$ space data structure. However, this operation, which takes $O(n^2)$ time to compute[1], is much more expensive than the basic cost of the star clustering algorithm, which is approximately $O(n)$ time. Thus, it is clear that the similarity matrix is a bottleneck. Computing this matrix is a one-time pre-processing operation. However, the data structure has to be available on a permanent basis. For these reasons, we now investigate several methods to improve on the similarity matrix bottleneck.

[1]Note that the actual time is $O(n^2)$ times the cost of a vector dot product; because the vectors are sparse, this translates into $O(n^2)$ with a high constant.

4.1 Sampled Stars

The first approximation algorithm uses sampling to compute the similarity matrix and is called the *sampled star* algorithm (see Figure 2). The basic idea behind this algorithm is to create a sample of the document collection that is much smaller than the actual collection. This sample can then be used to compute a complete Star Clustering, using the off-line star algorithm. For this small set, the computation of the similarity matrix is much faster. Finally, the rest of the documents can be inserted in the resulting clusters fast by comparing each document against the existing star centers only. Documents that are not close enough to any existing star centers (that is, all distances to existing star centers are below the threshold) form new clusters. Alternatively, the additional documents can be inserted in the cluster structure using the on-line star algorithm.

4.2 Linear-space Stars

For any threshold σ:

1. Let D be a set of n documents, p a desired probability, and σ a threshold.
2. Let $C = \emptyset$ denote the desired clustering.
3. Select a sample S of pairs of documents (d_1, d_2) from D
4. For each pair (d_1, d_2) in S if the dot product between $(d_1, d_2) > \sigma$ increase the degrees of d_1 and d_2.
5. Sort D in descending order by degree.
6. Find and mark all the star centers by examining one-by-one the sorted D.
7. For $i = 1$ to n insert d_i into all possible star centers.

Figure 3: The linear space sampled star algorithm.

The sampled star algorithm provides a more effective way to compute the overall clustering of a document set but even this algorithm requires the computation of a complete similarity matrix (which is smaller than the original matrix). An additional optimization is to remove entirely the similarity matrix. The key information used by the star algorithm is the degree of the nodes in the thresholded similarity graph. This information can be represented in an array. A trivial algorithm for generating the array is to compare every document against every other document and count the number of vector products about the threshold. Note that this method reduces significantly the space requirements but still necessitates $O(n^2)$ time to generate, where n is the number of documents. An alternative is to compute the vertex degrees approximately, using sampling. For each document, we first generate a sample of documents to be used for comparison. A dot product is computed between the document and each member of the sample set. The degree of the document vertex is given by the number of dot products that are above the threshold. Figure 3 summarizes this algorithm.

4.3 Distributed Stars

Another bottleneck for the star algorithm comes up in Internet applications, such as organizing data collected from various sites and databases by topic. Consider a task in which several databases are

For any threshold σ:

1. Let D be a set of n documents. Divide D into k disjoint sets $D_1 \dots D_k$.
2. Run the Star algorithm on k separate machines to produce the star clusterings $C_1 \dots C_k$.
3. Let $c_1 ... c_j$ be the set of star centers in all the star covers.
4. Run the Star algorithm on the set of documents $c_1 ... c_j$.
5. If two star centers are placed in the same cluster in the previous step, merge their clusters using a union operation.

Figure 4: The distributed star algorithm.

queried with the same question. The documents returned by these queries are to be fused and presented to the user in a coherent picture. One approach is to run the queries, download all documents, and organize the entire collection at the user site using the star algorithm. An alternative approach is to run the queries, organize the search results at the location of the database, and then merge these results on the user machine. This second alternative has several advantages: (1) the star algorithm can be run in parallel, which provides a speedup; (2) the document transfer operation can also be parallelized[2]; and (3) the local topic organizations can be viewed as a way of compressing the documents, can be used to generate the merged topics in the distributed collection, and can be transfered much faster than the actual documents to the user's machine.

For these reasons, we describe a third approximation of the star algorithm called *the distributed star* algorithm, which is useful especially when the document collection is very large. The distributed star algorithm provides parallelism and is based on a "divide and conquer" approach. The document collection is partitioned into several disjoint sets. The sets are clustered separately and the resulting clusters are then merged. Figure 4 shows the details of this algorithm. Note that for this version of the algorithm, the off-line Star algorithm can be replaced with the Sampled Star algorithm or with the Linear Space Star algorithm.

4.4 Experiments and Evaluations

We devised two experiments for the purpose of testing our algorithms on real-world data. Because we were limited by computer memory, we focused the experiments on the Linear Space Sampled Star algorithm (see Figure 3) which was introduced to optimize both time performance and space requirements.

In our first experiment, we ran the Linear Space Sampled Star algorithm on a 50000- document subset of the TREC volume 1 corpus at various sample sizes. We compared the output of the Linear Space Sampled Star algorithm with sampling to its output without sampling, and show these results in Figure 5. Note that when sampling is not used, the the Linear Space Sampled Star algorithm produces the same output as the Star algorithm.

To measure the difference between the outputs of the two algorithms, we calculated an aggregate precision and recall for each sample size as follows. For each cluster x in the output of the sampled algorithm, we calculated the precision and recall of the documents in x against each cluster in the output of the unsampled algorithm. We then determined the cluster y in the output of the unsampled algorithm that

[2]Note that if the number of documents is large and the network bandwidth is low, the cost of the transfer can be overwhelming. For example, (Rus et al, 1997) quantify experimentally the cost of transferring data over congested networks.

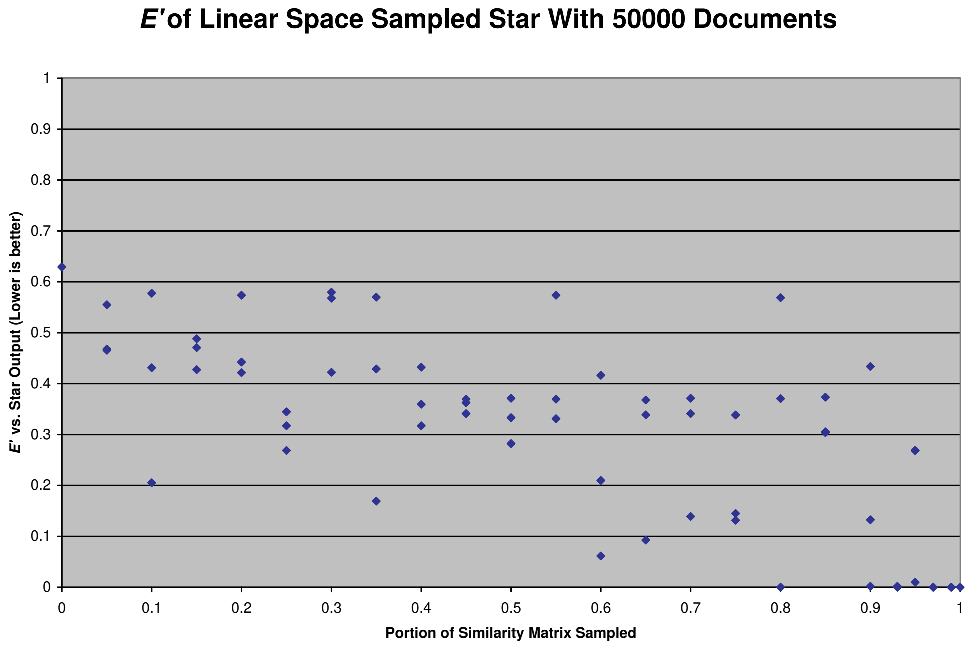

Figure 5: The effects of the sample size on the quality of clusters obtained using the Linear Space Sampled Star algorithm. The x-axis shows the sample size. The y-axis shows the aggregate E-measure computed relative to the star algorithm. The smaller the E-value is, the better the performance is. The experiment was done with a TREC subset of 50000 documents.

minimizes van Rijsbergen's (Rijsbergen, 1979) evaluation measure

$$E(p, r) = 1 - \frac{2}{1/p + 1/r}$$

where p and r are the standard *precision* and *recall* of the cluster with respect to the set of documents relevant to the topic. Finally, we calculated a weighted average E' of the E-values calculated previously, weighting each E value by the number of documents in the associated cluster. Figure 1 shows the results of this analysis. With larger samples, the sampled algorithm generally produced the exact same results as the algorithm that did not use sampling. As the portion of the similarity matrix sampled decreased, the results of the sampled algorithm deviated increasingly from those of the unsampled algorithm.

Our subsequent analyses sought to determine whether the divergent output of the sampled algorithm was inferior to the output of the unsampled algorithm. The original purpose of the Star algorithm was to calculate a cover of the input documents using as few star-shaped clusters as possible (Aslam et al, 1998; Aslam et al, 1999). The Linear Space Sampled Star algorithm also generates a cover of the input documents with star-shaped clusters, so we compared the number of clusters in the algorithm's output at varying sample sizes to the number of clusters in the output of the unsampled algorithm (see Figure 6). Surprisingly, even with samples as small as 5%, the number of clusters output by the sampled algorithm was never more than five percent larger than the number of clusters that the unsampled algorithm generated. In fact, the sampled algorithm generally covered the corpus with fewer star-shaped clusters than unsampled algorithm did.

Our second experiment compared the output of the Linear Space Sampled Star algorithm against categorization decisions made by humans. Specifically, the algorithm was run on 4925 documents from the FBIS corpus that had been labeled by humans with one or more of 47 different categories. We repeated the precision/recall analysis of the first experiment, using the 47 categories in the place of the output of the unsampled Star algorithm. As with the previous experiment, samples as small as 1% produced results

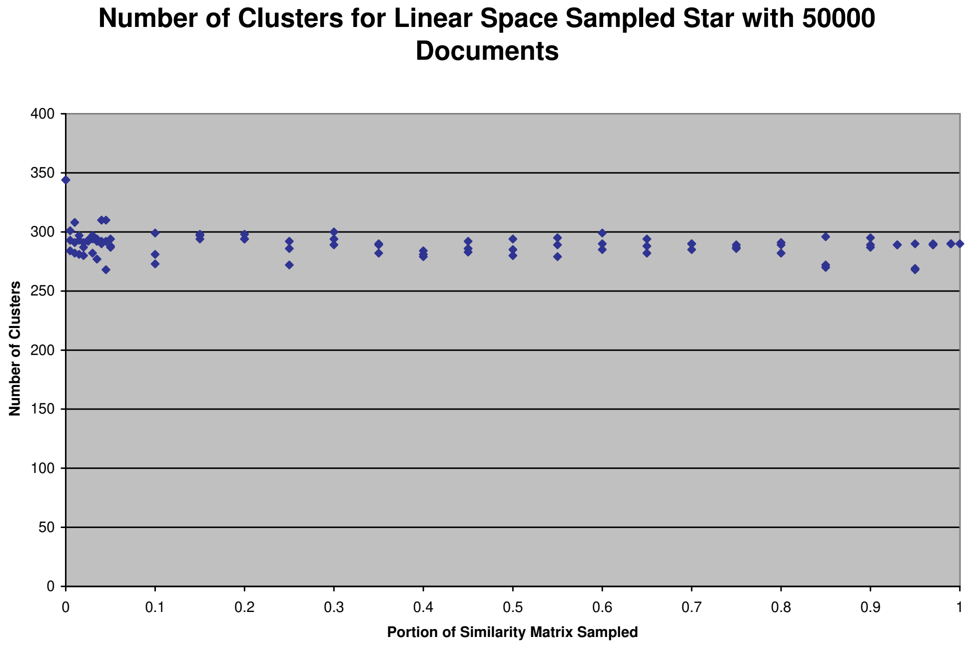

Figure 6: The effect of sampling on the number of clusters generates. The x-axis shows the sampling size. The y-axis shows the ratio between the number of clusters generated by the Linear Space Sampled Star algorithm to the number of clusters generated by the Star algorithm. We observe that sampling does not affect much the number of clusters discovered in the collection. The experiment was done with a TREC subset of 50000 documents.

comparable to a 100% sample (See Figure 7).

Overall, our experiments indicated that the Linear Space Sampled Star algorithm generates output comparable in quality to that of the Star algorithm, but uses considerably fewer CPU and memory resources. Both of our implementations of the Linear Space Star algorithm required only 81 megabytes of memory to process 50000 documents, 73 megabytes of which was only used to store the vector representations of the documents. On the other hand, an implementation of the Star algorithm that uses a sparse thresholded similarity matrix would require approximately 2.5 gigabytes of memory for 50000 documents, and a complete similarity matrix stored in a double-precision floating-point array would require 18.6 gigabytes of memory. The gains in performance due to sampling were similarly significant. Figure 8 shows the amount of time that the Linear Space Sampled Star algorithm requires to process 50000 documents at varying sample sizes. These times were measured on a 250 MHz. MIPS R10000 and do not include the time required to parse the documents. We found the running time of the algorithm to be almost directly proportional to the size of the sample. At sample sizes of less than 5%, the Linear Space Sampled Star algorithm organized documents at an average rate comparable to the bandwidth of most Internet connections (See Figure 9). Tests comparing the Star algorithm with the Linear Space Sampled Star algorithm on smaller data sets indicated that the overhead of sampling and reducing memory requirements result in an increase in running time of less than 5%.

Finally, we have conducted a small experiment on 1000 TREC documents to study the performance of the Distributed Star algorithm Figure 10 shows the accuracy of the distributed star algorithm relative to the off-line star algorithm. We note that when the number of computers is the same as the number of documents, Step 4 of the Distributed Star Algorithm (Figure 4) performs a star clustering of the entire collection. The same is true when there is a single machine. The greatest degree of parallelism and distribution is achieved when the number of machines is $\sqrt{m}$, where m is the number of machines in the system. For this experiment, $m = 1000$ and $\sqrt{m}$ is approximately 32. The experiment shows that the

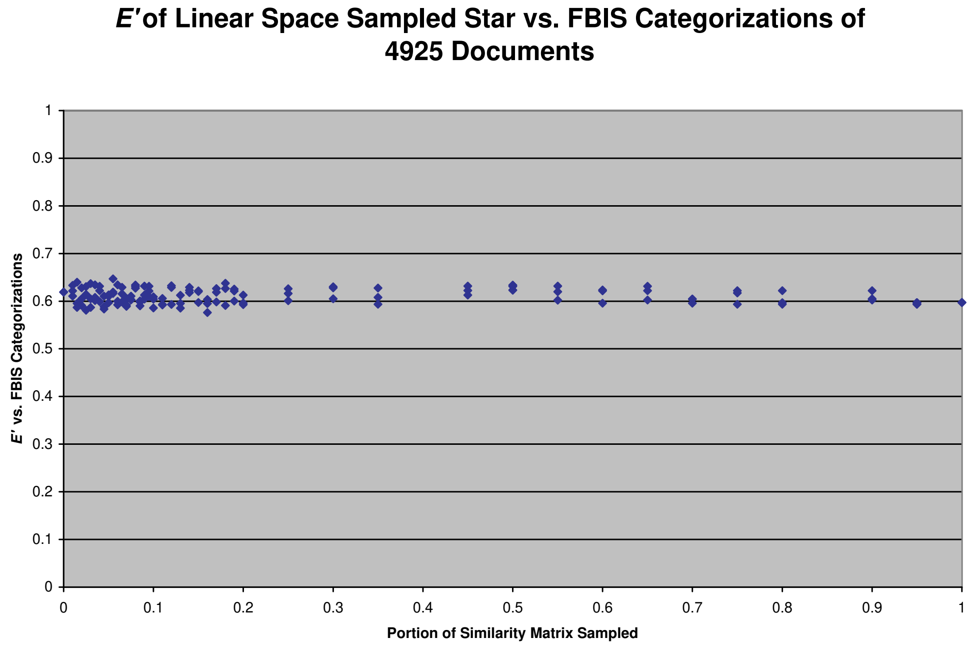

Figure 7: The effect of sampling on the quality of the clustering for the FBIS collection. The x-axis show the sampling size. The y axis shows the E-measure computed relative to the human clustering.

E-measure for 32 machines is about 41 %.

5 Conclusion

We presented a scalable algorithm for information organization. Scalability is a very important property for information organization algorithms especially when the collections are dynamic and Web-based. We implemented these algorithms as a scalable system for information organization. In the near future, we plan to expand our experimental collection to demonstrate the performance of our algorithms when dealing with hundreds of thousands of documents.

References

Aslam, J., K. Pelekhov, and D. Rus. (1998). Static and Dynamic Information Organization with Star Clusters. In *Proceedings of the 1998 Conference on Information Knowledge Management*, Baltimore, MD.

Aslam, J., K. Pelekhov, and D. Rus. (1999). A practical Clustering Algorithm for Static and Dynamic Information Organization. In *Proceedings of the 1999 Symposium on Discrete Algorithms*, Baltimore, MD.

Charikar, M., C. Chekuri, T. Feder, and R. Motwani. (1997). Incremental clustering and dynamic information retrieval, in *Proceedings of the* 29^{th} *Symposium on Theory of Computing*.

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

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

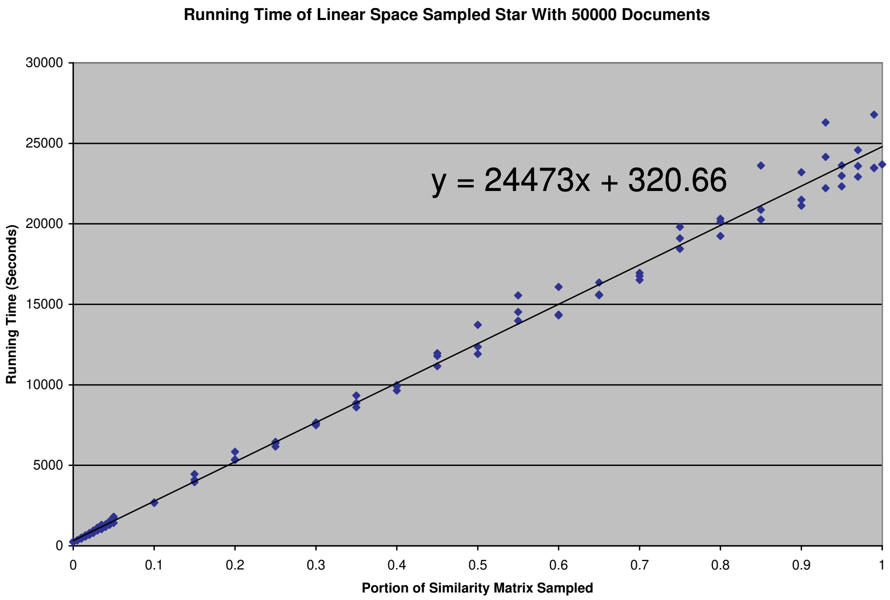

Figure 8: The running time of the Linear Sampled Star algorithm on a 50000 document subset of TREC. The x-axis shows the sample size and the y axis shows the running time in seconds.

Everitt, B. (1993). *Cluster Analysis.* Arnold, London.

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

Jain, A. and R. Dubes. (1988). *Algorithms for Clustering Data*, Prentice Hall.

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

Mirkin, B. (1996). *Mathematical classification and clustering.* Kluwer Academic Publishers, Boston.

van Rijsbergen, C.J.. (1979). *Information Retrieval*. Butterworths, London.

Rus, D., R. Gray, and D. Kotz. (1997). Transportable Information Agents. *Journal of Intelligent Information Systems*, vol 9. pp 215-238.

Sibson, P. (1973). SLINK: an optimally efficient algorithm for the single link cluster method. *Computer Journal* 16, pp30–34.

Silverstein, C. and J. Pedersen. (1997) Almost-Constant-Time Clustering of Arbitrary Corpus Subsets. In *Proceedings of SIGIR*, pp 60–66.

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

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

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

Zuckerman, D. (1993). 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.

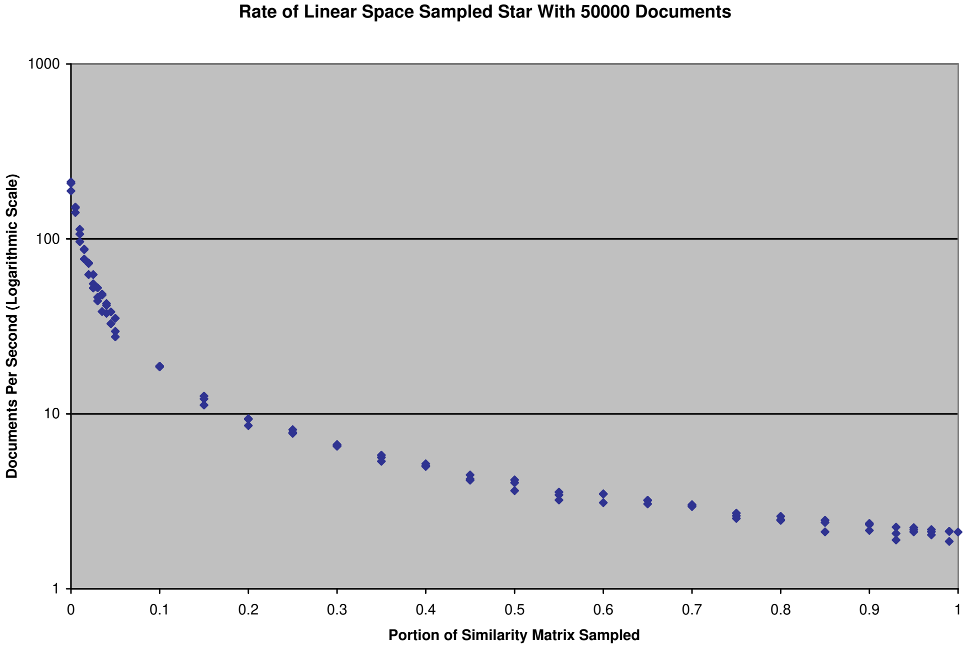

Figure 9: The effect of the sample size on the rate of the Linear Space Sampled Star algorithm (plotted on a logarithmic scale).

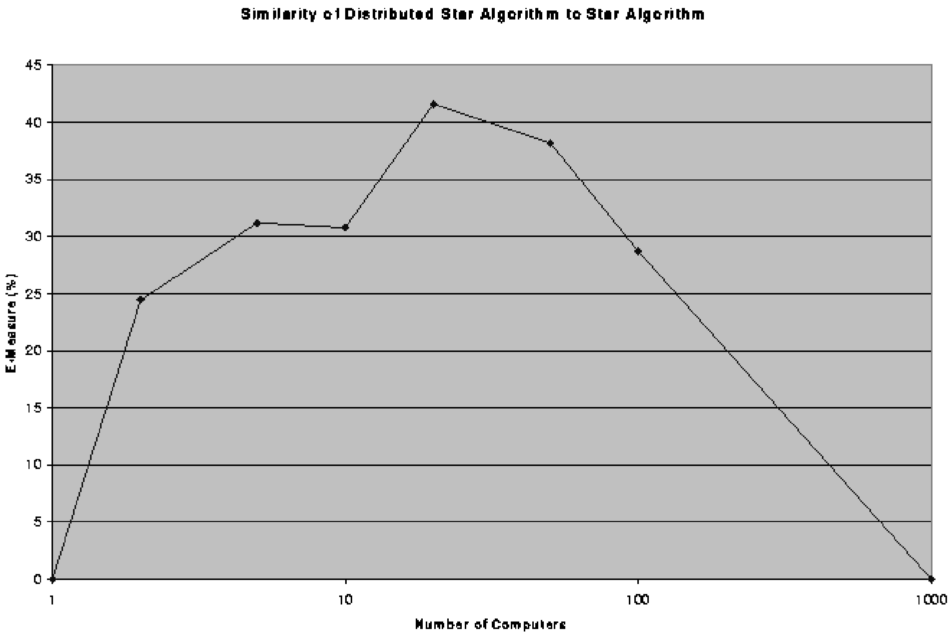

Figure 10: This graph shows the E-measure of the distributed star algorithm relative to the off-line star clustering of the same document set.