Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections

Douglass R. Cutting[1] David R. Karger[1,2] Jan O. Pedersen[1] John W. Tukey[1,3]

Abstract

Document clustering has not been well received as an information retrieval tool. Objections to its use fall into two main categories: first, that clustering is too slow for large corpora (with running time often quadratic in the number of documents); and second, that clustering does not appreciably improve retrieval.

We argue that these problems arise only when clustering is used in an attempt to improve conventional search techniques. However, looking at clustering as an information access tool in its own right obviates these objections, and provides a powerful new access paradigm. We present a document browsing technique that employs document clustering as its primary operation. We also present fast (linear time) clustering algorithms which support this interactive browsing paradigm.

1 Introduction

Document clustering has been extensively investigated as a methodology for improving document search and retrieval (see [15] for an excellent review). The general assumption is that mutually similar documents will tend to be relevant to the same queries, and, hence, that automatic determination of groups of such documents can improve recall by effectively broadening a search request (see [11] for a discussion of the cluster hypothesis). Typically a fixed corpus of documents is clustered either into an exhaustive partition, disjoint or otherwise, or into a hierarchical tree structure (see, for example, [8, 13, 2]). In the case of a partition, queries are matched against clusters and the contents of the best scoring clusters are returned as a result, possibly sorted by score. In the case of a hierarchy, queries are processed downward, always taking the highest scoring branch, until some stopping condition is achieved. The subtree at that point is then returned as a result. Hybrid strategies are also available.

These strategies are essentially variations of near-neighbor search[1] where nearness is defined in terms of the pairwise document similarity measure used to generate the clustering. Indeed, cluster search techniques are typically compared to direct near-neighbor search [9], and are evaluated in terms of precision and recall. Various studies indicate that cluster search strategies are not markedly superior to near-neighbor search, and, in some situations, can be inferior (see, for example, [6, 12, 4]). Furthermore, document clustering algorithms are often slow, with quadratic running times. It is therefore unsurprising that cluster search, with its indifferent performance, has not gained wide popularity.

Document clustering has also been studied as a method for accelerating near-neighbor search, but the development of fast algorithms for near-neighbor search has decreased interest in that possibility [1].

In this paper, we take a new approach to document clustering. Rather than dismissing document clustering as a poor tool for enhancing near-neighbor search, we ask how clustering can be effective as an access method in its own right. We describe a document browsing method, called *Scatter/Gather*, which uses document clustering as its primitive operation. This technique is directed towards information access with non-specific goals and serves as a complement to more focused techniques.

To implement Scatter/Gather, fast document clustering is a necessity. We introduce two new near linear time clustering algorithms which experimentation has shown to be effective, and also discuss reasons for their effectiveness.

1.1 Browsing vs Search

The standard formulation of the information access problem presumes a query, the user's expression of an information need. The task is then to search a corpus for documents that match this need. However, it is not difficult to imagine a situation in which it is hard, if not impossible, to formulate such a query precisely. For example, the

[1] Xerox Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304
[2] Stanford University
[3] Princeton University

15th Ann Int'l SIGIR'92/Denmark-6/92

[1] Also known as "vector space" or "similarity" search.

user may not be familiar with the vocabulary appropriate for describing a topic of interest, or may not wish to commit himself to a particular choice of words. Indeed, the user may not be looking for anything specific at all, but rather may wish to discover the general information content of the corpus. Access to a document collection in fact covers an entire spectrum: at one end is a narrowly specified *search* for a particular document, given something as specific as its title; at the other end is a *browsing* session with no well defined goal, satisfying a need to learn more about the document collection. It is common for a session to move across the spectrum, from browsing to search: the user starts with a partially defined goal which is refined as he finds out more about the document collection. Standard information access techniques tend to emphasize the search end of the spectrum. A glaring example of this emphasis is cluster search, where clustering, a technology capable of topic extraction, is submerged from view and used only to assist near-neighbor search.

We propose an alternative application for clustering in information access, taking our inspiration from the access methods typically provided with a conventional textbook. If one has a specific question in mind, and specific terms which define that question, one consults the index, which directs one to passages of interest. However, if one is simply interested in gaining an overview, or has a general question, one peruses the table of contents, which lays out the logical structure of the text. The table of contents gives a sense of what sort of questions might be answered by a more intensive examination of the text, and may also lead to specific sections of interest. One can easily alternate between browsing the table of contents and searching the index.

By direct analogy, we propose an information access system with two components: our browsing method, Scatter/Gather, which uses a cluster-based, dynamic table-of-contents metaphor for navigating a collection of documents; and one or more word-based, directed, text search methods, such as near-neighbor search or snippet search [7]. The browsing component describes groups of similar documents, one or more of which can be selected for further examination. This can be iterated until the user is directly viewing individual documents. Based on documents found in this process, or on terms used to describe document groups, the user may, at any time, switch to a more focused search method. In particular, we anticipate that the browsing tool will not necessarily be used to find particular documents, but may instead help the user formulate a search request, which will then be serviced by some other means. Scatter/Gather may also be used to organize the results of word-based queries that retrieve too many documents.

2 Scatter/Gather Browsing

In the basic iteration of the proposed browsing method, the user is presented with short summaries of a small number of document groups.

Initially the system *scatters* the collection into a small number of document groups, or *clusters*, and presents short summaries of them to the user. Based on these summaries, the user selects one or more of the groups for further study. The selected groups are *gathered* together to form a subcollection. The system then applies clustering again to scatter the new subcollection into a small number of document groups, which are again presented to the user. With each successive iteration the groups become smaller, and therefore more detailed. Ultimately, when the groups become small enough, this process bottoms out by enumerating individual documents.

2.1 An Illustration

We now describe a Scatter/Gather session, where the text collection consists of about 5000 articles posted to the *New York Times News Service* during the month of August 1990. This session is summarized in figure 1. Here, to simplify the figure, we manually assigned single-word labels based on the full cluster descriptions. The full session is provided as Appendix A.

Suppose the user wants to find out what happened that month. Several issues prevent the application of conventional search techniques:

- The information need is too vague to be described as a single topic.
- Even if a topic were available, the words used to describe it may not be known to the user.
- The words used to describe a topic may not be those used to discuss the topic and may thus fail to appear in articles of interest. For example, articles concerning international events need never use the words "international event".
- Even if some words used in discussion of the topic were available, documents may fail to use precisely those words, *e.g.,* synonyms may be used instead.

With Scatter/Gather, rather than being forced to provide terms, the user is presented with a set of clusters, an outline of the corpus. She need only select those clusters which seem potentially relevant to the topic of interest. In the example, the big stories of the month are immediately obvious from the initial scattering: Iraq invades Kuwait, and Germany considers reunification. This leads the user to focus on international issues: she selects the 'Kuwait' and 'Germany' and 'Oil' clusters. These three clusters are gathered together.

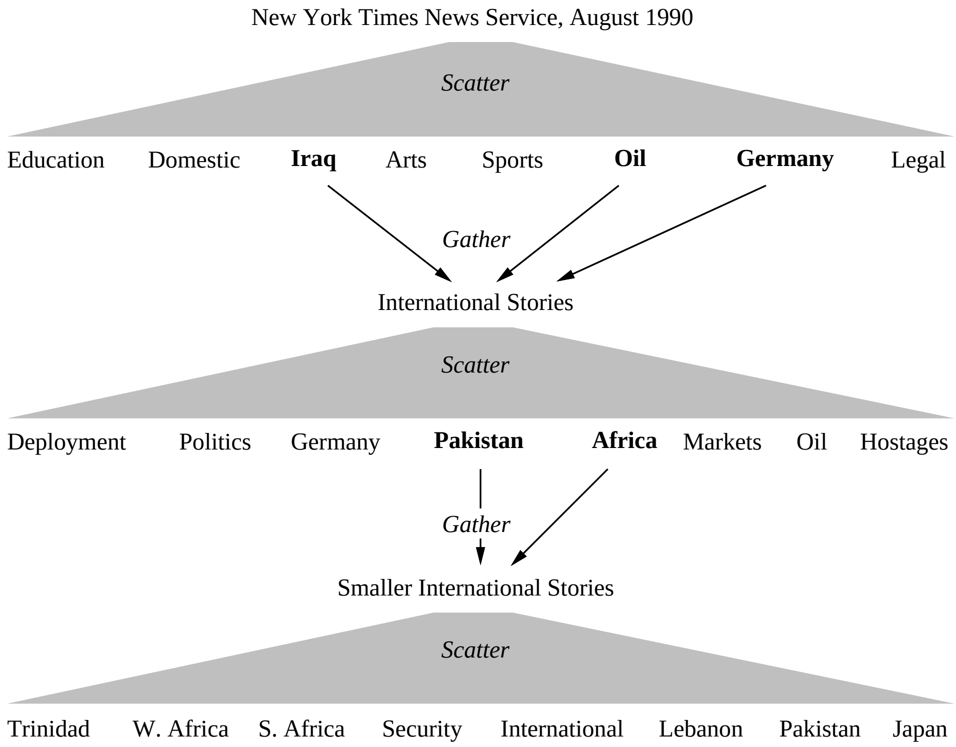

Figure 1: Illustration of Scatter/Gather

This reduced corpus is then reclustered on the fly to produce eight new clusters covering the reduced corpus. Since the reduced corpus contains a subset of the articles, these new clusters reveal a finer level of detail than the original eight. The articles on the Iraqi invasion and some of the 'Oil' articles have now been separated into clusters discussing the U.S. military deployment, the effects of the invasion upon the oil market, and one which the user deduces is about hostages in Kuwait.

The user feels her understanding of these large stories is adequate, but wishes to find out what happened in other corners of the world. She selects the 'Pakistan' cluster, which also contains other foreign political stories, and a cluster containing articles about Africa. This reveals a number of specific international situations as well as a small collection of miscellaneous international articles. The user thus learns of a coup in Pakistan, and about hostages being taken in Trinidad, stories otherwise lost among the major stories of that month.

2.2 Requirements

Scatter/Gather depends on the existence of two facilities. First, since clustering and reclustering is an essential part of the basic iteration, we need an algorithm which can appropriately cluster a large number of documents within a time tolerable for user interaction (*e.g.,* less than a minute). Second, given a group of documents, some method for automatically summarizing that group must be specified. This cluster description must be sufficiently revealing for to give the user a sense of the topic defined by the group, yet short enough for many descriptions to be appreciated simultaneously.

We will present two algorithms that meet the first requirement. *Buckshot* is a fast clustering algorithm suitable for the online reclustering essential for Scatter/Gather. *Fractionation* is another, more careful, clustering algorithm whose greater accuracy makes it suitable for the static offline partitioning of the entire corpus which is presented first to the user. We also define the *cluster digest,* an easy to generate, concise description of a cluster suitable for Scatter/Gather.

3 Document Clustering

Before presenting our document clustering algorithms, we review the terminology established in prior work, and discuss why existing clustering algorithms fail to meet our needs. Throughout this paper, n denotes the number of documents in the collection, and k denotes the desired number of clusters.

In order to cluster documents one must first establish a pairwise measure of document similarity and then define a method to partition the collection into clusters of similar documents. Numerous document similarity measures have been proposed, all of which treat each document as a set of words, often with frequency information, and measure the degree of word overlap between documents [11]. The documents are typically represented by sparse vectors of length equal to the number of unique words (or types) in the corpus. Each component of the vector has a value reflecting the occurrence of the corresponding word in the document. We may use a binary scale, in which the value is one or zero, to represent the presence or the absence of the word, or the value might be some function of the word's frequency within that document. If a word does not occur in a document, its value is zero. A popular similarity measure, the *cosine measure*, computes the cosine of the angle between these two sparse vectors. If both document vectors are normalized to unit length, the cosine is, of course, simply the inner product of the two vectors. Other measures include the Dice and Jaccard coefficients, which are normalized word overlap counts. Willett [15] has suggested that that the choice of similarity measure has less qualitative impact on clustering results than the choice of clustering algorithm.

Two different types of document clusters can be constructed. One is a *flat partition* of the documents into a collection of subsets. The other is a *hierarchical cluster*, which can be defined recursively as either an individual document or a partition of the corpus into sets, each of which is then hierarchically clustered. A hierarchical clustering defines a tree, called a *dendrogram*, on the documents.

Numerous clustering algorithms have been applied to build hierarchical document clusters, including, most prominently, single-linkage hierarchical clustering [5, 2]. These algorithms generally proceed by iteratively considering all pairs of clusters built so far, and fusing the pair which exhibits the greatest similarity into a single document group (which then becomes a node of the dendrogram). They differ in the procedure used to compute similarity when one of the pair is the product of a previous fusion. Single-linkage clustering defines the similarity as the maximum similarity between any two individuals, one from each of the two groups. Alternative methods consider the minimum similarity (complete-linkage), the average similarity (group-average linkage), as well as other aggregate measures. Although single-linkage clustering is known to have an undesirable chaining behavior, typically forming elongated straggly clusters, it remains popular due to its simplicity and the availability of an optimal space and time algorithm for its computation [10].

These algorithms share certain common characteristics. They are *agglomerative,* in that they proceed by iteratively choosing two document groups to agglomerate into a single document group. They agglomerate in a *greedy* manner, in that the pair of document groups chosen for agglomeration is the pair which is considered best or most similar under some criterion. Lastly, they are *global* in that all pairs of inter-group similarities are considered in the course of selecting an agglomeration. Global algorithms have running times which are intrinsically $\Omega(n^2)$,[2] because all pairs of similarities must be considered. This sharply limits their usefulness, even given algorithms that attain the theoretical quadratic lower bound on performance.

Partitional strategies, those that strive for a flat decomposition of the collection into sets of documents rather than a hierarchy of nested partitions, have also been studied [8, 13]. Some of these algorithms are global in nature and thus have the same slow performance as the above mentioned greedy, global, agglomerative algorithms. Other partitional algorithms, by contrast, typically have rectangular running times, *i.e.*, $O(kn)$. Generally, these algorithms proceed by choosing, in some manner, a number of *seeds* equal to the desired size (number of sets) of the final partition. Each document in the collection is then assigned to the closest seed. As a refinement, the procedure can be iterated, with, at each stage, an improved selection of cluster seeds. It is noteworthy that any partitional clustering algorithm can be transformed into a hierarchical clustering algorithm by recursively partitioning each of the clusters found in an application of the partitioning algorithm.

One application of a partitional clustering has been to improve the performance of near-neighbor search by including, with each document, some closely related documents that might otherwise be missed. However, to be useful for near-neighbor search, the partition must be fairly fine, since it is desirable for each set to only contain a few documents. For example, Willett generates a partition who size is related to the number of unique words in the document collection [13]. From this perspective, the potential computational benefits of a seed-based strategy are largely obviated by the large size (relative to the number of documents) of the required partition. For this reason partitional strategies have not been aggressively pursued by the information retrieval community.

We present two partitioning algorithms which use techniques drawn from the hierarchical algorithms, but which acheive rectangular time bounds. For our application, the number of clusters desired is small and thus the speedup over quadratic time algorithms is substantial.

[2] Willett [14] discusses an inverted file approach which can ameliorate this quadratic behavior when a large number of small clusters are desired. Unfortunately, when clusters are large enough to contain a large proportion of the terms in the corpus, this approach yields less improvement.

4 Definitions

For each document α in a collection (or corpus) C, let the *countfile* $c(\alpha)$ be the set of words, with their frequencies, that occur in that document.[3] Let V be the set of unique words occurring in C. Then $c(\alpha)$ can be represented as a vector of length $|V|$;

$$c(\alpha) = \{f(w_i, \alpha)\}_{i=1}^{|V|}$$

where w_i is the ith word in V and $f(w_i, \alpha)$ is the frequency of w_i in α.

To measure the similarity between pairs of documents, α and β, let us employ the cosine between monotone element-wise functions of $c(\alpha)$ and $c(\beta)$. In particular, let

$$s(\alpha, \beta) = \frac{\langle g(c(\alpha)), g(c(\beta))\rangle}{\|g(c(\alpha))\| \|g(c(\beta))\|}$$

where g is a monotone damping function, "$\langle \cdot, \cdot \rangle$" denotes inner product, and $\| \cdot \|$ denotes vector norm. It has been our experience that taking g to be component-wise square-root produces better results than the traditional component-wise logarithm.

It is useful to consider similarity to be a function of document *profiles* $p(\alpha)$, where

$$p(\alpha) = \frac{g(c(\alpha))}{\|g(c(\alpha))\|},$$

in which case

$$s(\alpha, \beta) = \langle p(\alpha), p(\beta)\rangle = \sum_{i=1}^{|V|} p(\alpha)_i \, p(\beta)_i.$$

Suppose Γ is a set of documents, or a *document group*. A simple profile can be associated with Γ by defining it to be the normalized sum of profiles of the contained individuals. Let

$$\hat{p}(\Gamma) = \sum_{\alpha \in \Gamma} p(\alpha)$$

be the unnormalized sum profile, and then

$$p(\Gamma) = \frac{\hat{p}(\Gamma)}{\|\hat{p}(\Gamma)\|}.$$

Similarly, the cosine measure can be extended to Γ by employing this profile definition:

$$s(\Gamma, x) = \langle p(\Gamma), p(x)\rangle.$$

Sometimes for our purposes, the normalized sum profile is not a good measure of a document group's "contents" because it takes into account documents which lie on the outskirts of the group. To solve this problem, we define the *trimmed sum profile* $p_m(\Gamma)$ for any cluster Γ by considering only the m "most central" documents of the cluster. For every α in Γ let $r_m(\Gamma)$ be the m documents α whose similarity to Γ, namely $s(\alpha, \Gamma)$, is largest. Then define

$$\hat{p}_m(\Gamma) = \sum_{\alpha \in r_m(\Gamma)} p(\alpha).$$

and

$$p_m(\Gamma) = \hat{p}_m(\Gamma)/\|\hat{p}_m(\Gamma)\|.$$

This computation can be completed in time proportional to $|\Gamma|$.[4] The trimming parameter m may be defined adaptively as some percentage of $|\Gamma|$, or may be fixed.

4.1 Cluster Digest

Another description of a document group is in some sense dual to the trimmed sum profile. Rather than considering the central documents of a cluster, we can consider the central *words*, namely those which appear most frequently in the group as a whole. We thus define $t_w(\Gamma)$, the *topical* words of Γ, to be the w highest weighted terms in $p(\Gamma)$ (or perhaps in $p_m(\Gamma)$).

Taken together, the two sets $(r_m(\Gamma), t_w(\Gamma))$ form the (m, w) *cluster digest* of Γ, a short description of the contents of the cluster. The cluster digest can easily be computed in time $O(|\Gamma| + |V|)$, and is in fact the summary used to describe a cluster to a user of Scatter/Gather.

5 Partitional Clustering

Seed-based partitional clustering algorithms have three phases:

1 Find k centers.

2 Assign each document in the collection to a center.

3 Refine the partition so constructed.

The result is a set P of k disjoint document groups such that $\bigcup_{\Pi \in P} \Pi = C$.

The Buckshot and Fractionation algorithms are both designed to find the initial centers. They can be thought of as rough clustering algorithms, however their output is only used to define centers. Both algorithms assume the existence of some algorithm which clusters well, but which may run slowly. Let us call this procedure the *cluster subroutine*. We use group average agglomerative clustering for this subroutine (see appendix B). Each of our algorithms uses this cluster subroutine locally over small sets, and builds on its results to find the k centers.

[3] Throughout this paper, lower case Greek letters will be used to denote individual documents. Upper case Greek letters will denote sets of documents (document groups) and upper case Roman letters will denote sets of document groups.

[4] A full sort of the similarities is not required.

Buckshot applies the cluster subroutine to a random sample to find centers. Fractionation uses successive application of the cluster subroutine over fixed sized groups to find centers. We believe that Fractionation is the more accurate center finding procedure. However, Buckshot is significantly faster, and, hence, is more appropriate for the on-the-fly online reclustering required by iterations of Scatter/Gather. Fractionation can be used to establish the primary partitioning of the entire corpus, which is displayed in the first iteration of Scatter/Gather.

We implement Step 2 by assigning each document to the "nearest" center (in a sense to be defined later).

Our refinement algorithms also reflect a time-accuracy tradeoff. The simplest refinement procedure, iterated move-to-nearest, is fast but limited. A more comprehensive refinement is achieved through repeated application of procedures that attempt to Split, Join, and clarify elements of the partition P.

5.1 Finding Initial Centers

Buckshot

The idea of the buckshot algorithm is quite simple. To achieve a rectangular time clustering algorithm, merely choose a small random sample of the documents (of size $\sqrt{kn}$), and apply the cluster subroutine. Return the centers of the clusters found. This algorithm clearly runs in time $O(kn)$.

Since random sampling is employed, the Buckshot algorithm is not deterministic. That is, repeated calls to this algorithm on the same corpus may produce different partitions, although in our experience repeated trials generally produce qualitatively similar partitions.

Fractionation

The Fractionation algorithm finds k centers by initially breaking C into N/m buckets of a fixed size $m > k$. The cluster subroutine is then applied to each of these buckets separately to agglomerate individuals into document groups such that the reduction in number (from individuals to groups in each bucket) is roughly a factor of ρ. These groups are now treated as if they were individuals, and the entire process repeated. The iteration terminates when only k groups remain. Fractionation can be viewed as building a $1/\rho$ branching tree bottom up, where the leaves are individual documents, terminating when only k roots remain.

Suppose the individuals in C are enumerated, so that $C = \alpha_1, \alpha_2, \ldots, \alpha_n$. This ordering could reflect an extrinsic ordering on C, but a better procedure sorts C based on a key which is the word index of the j^{th} most common word in each individual. Typically j is a small number, such as three, which favors medium frequency terms. This procedure thus encourages nearby individuals in the corpus ordering to have at least one word in common.

The initial bucketing creates a partition

$$B = \{\Theta_1, \Theta_2, \ldots, \Theta_{n/m}\}$$

such that

$$\Theta_i = \{\alpha_{m(i-1)+1}, \alpha_{m(i-1)+2}, \ldots, \alpha_{mi}\}.$$

Each Θ_i is then separately clustered (using the cluster subroutine) into ρm groups, where ρ is the desired reduction factor. Note that each of these computations occurs in m^2 time, and, hence, all n/m occur in nm time. Each application of agglomerative clustering produces an associated partition $R_i = \{\Phi_{i,1}, \Phi_{i,2}, \ldots, \Phi_{i,\rho m}\}$. The union of the documents groups contained in these partitions are then treated as individuals for the next iteration. That is, define

$$C' = \{\Phi_{i,j} : \ 1 \leq i \leq n/m, \ 1 \leq j \leq \rho m\}$$

C' inherits an enumeration order by taking the $\Phi_{i,j}$ in lexicographic order on i and j. The process is then repeated with C' replacing C. That is, the ρn components of C' are broken into $\rho n/m$ buckets, which are further reduced to $\rho^2 n$ groups through separate agglomeration. The process terminates at iteration j if $\rho^j n < k$. At this point one final application of agglomerative clustering can reduce the remaining groups to a partition P of size k.

To determine the running time, observe that the j^{th} iteration, which operates on $\rho^j n$ items, takes time $\rho^j nm$. The overall running time is thus $O(nm(1+\rho+\rho^2+\ldots)) = O(mn)$. Thus if $m = O(k)$ this algorithm has rectangular running time.

5.2 Assigning Documents to Centers

Once k centers have been found, and suitable profiles defined for those centers, each document in C must be assigned to one of those centers based on some criterion. The simplest algorithm, *Assign-to-Nearest*, assigns each document to the nearest center.

Let G be a partition of the collection into k groups, and let Γ_i be the ith group in G. Let $\alpha \in \Pi_i$ if i maximizes $s(\alpha \Gamma i)$. Ties can be broken by assigning α to the group with lowest index. The set $P = \{\Pi_i\}, \ 0 \leq i \leq k$ is then the desired partition.

P can be efficiently computed by constructing an inverted map for the k centers $p_m(\Gamma_i)$, and for each $\alpha \in C$ simultaneously computing the similarity to all the centers. In any case, the cost of this procedure is proportional to kn.

5.3 Refinement

Given an initial clustering, it is now desirable to refine it into a better one. As with our initial clustering algorithms, there is a tradeoff between speed and accuracy.

The simplest process is simply to iterate the Assign-to-Nearest process just discussed. The Split algorithm separates poorly defined clusters into two well separated parts and Join merges clusters which are too similar.

Iterated Assign-to-Nearest

The Assign-to-Nearest procedure mentioned above can also be seen as the first of our refinement algorithms. From a given set of clusters, we generate cluster centers using the trimmed sum profiles above, and we then assign each document to the nearest center so as to form new clusters. This process can be iterated indefinitely, though it makes its greatest gains in the first few steps, and hence is typically iterated only a small fixed number of times.[5]

Split

Split divides each document group Γ in a partition P into two new groups. This can be accomplished by applying Buckshot clustering (without refinement) with $C = \Gamma$ and $k = 2$. The resulting Buckshot partition G provides the two new groups.

Let $P = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_k\}$ and let $G_i = \{\Gamma_{i,1}, \Gamma_{i,2}\}$ be a two element Buckshot partition of Γ_i. The new partition P' is simply the union of the G_i's:

$$P' = \bigcup_{i=1}^{k} G_i.$$

Each application of Buckshot requires time proportional to $|\Gamma_i|$. Hence, the overall computation can be performed in time proportional to N.

A modification of this procedure would only split groups that score poorly on some coherency criterion. One simple criterion is the cluster self similarity $s(\Gamma, \Gamma)$. This quantity is in fact proportional to the average similarity between documents in the cluster, as well as to the average similarity of a document to the cluster centroid. We thus define:

$$A(\Gamma) = s(\Gamma, \Gamma).$$

Let $r(\Gamma_i, P)$ be the rank of $A(\Gamma_i)$ in the set

$$\{A(\Gamma_1), A(\Gamma_2), \ldots, A(\Gamma_k)\}.$$

The procedure would then only split groups such that $r(\Gamma, P) < \rho k$ for some ρ, $0 < \rho \leq 1$. This modification does not change the order of the algorithm since the coherence criterion can be computed in time proportional to N.

[5] Excessive iteration may in fact worsen the partition rather than improving it, since "fuzzy" elongated clusters can pull documents away from other clusters and become even fuzzier.

Join

The purpose of the Join refinement operator is to merge document groups in a partition P that are not usefully distinguished by their cluster digests. Since, by definition, any two elements of P are disjoint, they will never have "typical" documents in common. However, their lists of "topical" words may well overlap. Therefore the criterion of distinguishability between two groups Γ and Δ will be

$$T(\Gamma, \Delta) = |t_w(\Gamma) \cap t_w(\Delta)|$$

where $t_w(\Gamma)$ is the list of w most topical words for Γ. We merge Γ and Δ if $T(\Gamma, \Delta) > p$, for some p, $0 < p \leq w$.

Determining the topical words for each cluster takes time proportional to the number of words in the corpus, and we must then compute k^2 intersections to decide which clusters to merge. In large corpora, the number of words is typically less than the number of documents, and the running time of Join is thus $O(kn)$.

6 Application to Scatter/Gather

Combinations of the various initial clustering and refinement procedures give several possible complete clustering algorithms. We have used two of these combinations in the course of implementing the Scatter/Gather method.

The initial partition used in Scatter/Gather is completely determined by the corpus under consideration. Hence, when the corpus is available in advance, one can compute the initial partition offline. We can therefore use a slower clustering algorithm to improve the accuracy of the initial partition. However, for corpora consisting of tens of thousands of documents, a quadratic time algorithm is likely to be too slow even for offline computation. We thus use the Fractionation algorithm to find centers, and then perform a great deal of refinement using the Split, Join, and Assign-to-Nearest operators. Note that the running time for each of the refinement procedures is $O(kN)$ and thus does not affect the overall running time.

In an interactive session, however, it is vital for the clustering algorithm to run as quickly as possible, even at the expense of some accuracy. We therefore use the Buckshot center finding procedure, and then follow it with a bare minimum of refinement. We have found that two iterations of the Assign-to-Nearest procedure yield a reasonably accurate clustering, and that further refinement produces additional improvement, but with quickly diminishing returns.

By virtue of the Buckshot center finding procedure this algorithm is not deterministic. However, in the contemplated application, Scatter/Gather, it is more important that the partition be computed at high speed than that the algorithm be deterministic. Indeed, the lack of determinism might be interpreted as a feature, since the user then has the option of discarding an unrevealing partition in favor of a fresh reclustering.

The overall complexity of both clustering procedures described in this section is clearly $O(kN)$. The constant factor for the Buckshot-based procedure is small enough to permit interactive use with large document collections. The Fractionation-based procedure has a somewhat larger constant factor, but one which is still acceptable for offline applications.

6.1 Naturally Clustered Data

It is worth examining the performance of our algorithms when the data set consists of well separated clusters of points. If the input data has k natural clusters, *i.e.*, the smallest intra-cluster document similarity is larger than the largest inter-cluster document similarity, then both of our algorithms will find this partition.

For Buckshot, if we have a corpus containing k widely separated and equal size centers, then a random sample of size $\sqrt{kn}$ will select some documents from each of the centers with high probability so long as $n \gg k \ln k$. This will certainly be true for our case in which $k = 20$ or so. To see this, compute the probability that, if we choose a sample of size s, we fail to get any individual from some cluster. This is at most k times the probability that none of our s individuals is a member of cluster i, namely $(1 - 1/k)^s$. So, the total probability of failure is at most $k(1 - 1/k)^s$. If we now take $s = ak \ln k$ for some a, then the failure probability is at most

$$k(1 - 1/k)^{ak \ln k} \leq k^{1-a}.$$

Thus in our case, with $k = 20$, taking $a = 5$ means that 400 samples find all the clusters 999 times in 1000. Given that we start with at least one element from each cluster, our resulting clusters will each be a subset of one of the clusters. Thus the set of centers found will include a center within each actual cluster.

For Fractionation, we need merely note that if we have more than k documents in a single bucket, some pair of them is necessarily in the same actual cluster. Then clearly, this pair will be merged in preference to any other pair. Therefore, no pair of documents not in the same cluster will ever be merged. Thus, when we finish, each cluster we have found will be a subset of some one of the actual clusters.

7 Conclusion

Scatter/Gather demonstrates that document clustering can be an effective information access tool in its own right. The table-of-contents metaphor gives the method an intuitive basis, and experience has shown that it is indeed easy to use. Scatter/Gather is particularly helpful in situations in which it is difficult or undesirable to specify a query formally. Claims of improved performance must await evaluation metrics appropriate to the vaguely defined information access goals in which Scatter/Gather excels.

To support Scatter/Gather, fast clustering algorithms are essential. Clustering can be done quickly by working in a local manner on small groups of documents rather than trying to deal with the entire corpus globally.

For extremely large corpora, even the linear time clustering achieved by the Buckshot or Fractionation algorithms may be too slow. We are working to develop variations on Scatter/Gather which will scale to arbitrarily large corpora, under the assumption that linear time preprocessing will always be feasible.

Clearly, the accuracy of the Buckshot and Fractionation algorithms is affected by the quality of the clustering provided by the slow cluster subroutine. This provides further motivation to find highly accurate clustering algorithms, whatever their running time may be.

A A Scatter/Gather Session

In figures 2 through 5, we present the full output of the Scatter/Gather session described in section 2.1. The corpus is the set of articles distributed by the *New York Times News Service* during the month of August 1990. This consists of roughly 30 megabytes of ASCII text in about 5000 articles. Some articles are repeated due to updates of news stories.

Here our goal is to learn about international political events during this month. To create the initial partition we've applied the Buckshot clustering algorithm (figure 2). Fractionation is recommended for this task, time permitting.

Each cluster is described with the two line display of its cluster digest. The first line contains the number of the cluster, the number of documents in the cluster, and titles of documents near the centroid. The second line contains words frequent in the cluster.

We select clusters 2 (Iraq's invasion of Kuwait), 5 (Markets, including oil) and 6 (Germany, and probably other international issues) as those which seem likely to contain articles of interest, recluster, and display a new cluster digest (figure 3).

Next, in figure 4, we iterate, this time selecting clusters 3 (Pakistan, and probably other international issues) and 4 (African issues). Specific incidents have been separated out. We find hostages in Trinidad, war in Liberia, police action in South Africa, and so on.

We obtain more detail about the situation in Liberia by viewing the titles of the articles contained in that cluster (figure 5).

```
> (time (setq first (outline (all-docs tdb))))
cluster 4970 items
global cluster 199 items...sizes: 18 24 53 5 25 47 13 14
move to nearest...sizes: 517 1293 835 86 677 1020 273 269
move to nearest...sizes: 287 1731 749 275 481 844 310 293
0 (287) CRITICS URGE NEW METHODS; PROGRAMS FOR PARENTS THE; TEACHING SUBJECTS T
 school, year, student, child, university, state, program, percent, study, educ
1 (1731) FEDERAL WORK PROGRAMS HE; RESORT TAKES STEPS TO PR; AMERICANS CUT BACK
 year, state, york, city, million, day, service, company, week, official, house
2 (749) PENTAGON SAYS 60,000 IRA; BUSH ''DRAWS A LINE'' IN; BUSH SAYS FOREIGNER
 iraq, iraqi, kuwait, american, state, unite, saudi, official, military, presid
3 (275) Trillin's Many Hats; New Musical from the cre; After Nasty Teen-Agers i
 film, year, music, play, company, movie, art, angeles, york, american, directo
4 (481) TWISTS AND TURNS MAY MEA; SAX LOOKING FOR RELIEF I; PAINTING THE DODGER
 game, year, play, team, season, win, player, day, league, hit, right, coach, l
5 (844) CRISIS PUSHES OIL PRICES; WHY MAJOR PANIC OVER A M; OIL PRICES RISE AS
 price, oil, percent, market, company, year, million, stock, day, rate, week, s
6 (310) LEADERS OF TWO GERMANYS ; REPRESENTATIVES OF TWO G; SECURITY COUNCIL RE
 government, year, state, party, political, country, official, leader, presiden
7 (293) U.S. APPEALS ORDER FREEI; DID JUDGE MOVE TOO HASTI; MAYOR BARRY CONVICT
 case, court, charge, year, judge, lawyer, attorney, trial, jury, federal, dist
real time  131258 msec
```

Figure 2: Initial Scattering

```
> (time (setq second (outline first 2 5 6)))
cluster 1903 items
global cluster 123 items...sizes: 51 8 5 5 4 7 28 15
move to nearest...sizes: 730 67 65 62 56 99 714 110
move to nearest...sizes: 650 66 57 117 59 242 586 126
0 (650) PENTAGON SAYS 60,000 IRA; BUSH SAYS FOREIGNERS DET; BUSH ''DRAWS A LINE
 iraq, iraqi, american, kuwait, state, unite, military, official, president, sa
1 (66) LEGISLATIVE LEADERS BACK; THE PROBLEM WITH AN EARL; ROAD STILL TOUGH FOR
 party, state, election, year, political, candidate, vote, campaign, democratic
2 (57) IN PUSH FOR UNIFICATION,; IN PUSH FOR UNIFICATION,; LEADERS OF TWO GERMA
 german, east, germany, west, year, government, soviet, union, state, unificati
3 (117) BHUTTO GOVERNMENT DISMIS; IN FRACTIOUS PAKISTAN, G; PAKISTANIS FEEL LET
 government, minister, year, party, political, military, country, official, sta
4 (59) DEATH TOLL EXCEEDS 500 I; DE KLERK, MANDELA HOLD U; NEGOTIATIONS TO SETT
 african, government, south, leader, police, national, fight, group, official,
5 (242) WEST GERMANS TO BUY FIRE; FIRST EXECUTIVE CORP. EA; FARM BANK, MERRILL
 company, million, percent, share, year, corp, stock, market, sell, price, pres
6 (586) OIL PRICES RISE AS STOCK; MIDEAST CRISIS PUSHES OI; WHY MAJOR PANIC OVE
 oil, price, percent, market, year, company, day, stock, million, rate, week, f
7 (126) IRAQ GRANTS 237 FOREIGN ; WOMAN TELLS OF 12 DAYS I; CONCERN HEIGHTENS F
 kuwait, iraqi, american, iraq, saudi, day, year, invasion, country, state, ara
real time  54184 msec
```

Figure 3: Second Scatter

```
> (time (setq third (outline second 3 4)))
cluster 176 items
global cluster 37 items...sizes: 1 4 12 1 5 3 8 3
move to nearest...sizes: 4 16 44 1 23 7 71 10
move to nearest...sizes: 5 16 28 1 51 7 55 13
0 (5) MUSLIM MILITANTS LAY DOW; MUSLIM MILITANTS LAY DOW; DRAMA IS OVER BUT BOO
 government, trinidad, minister, parliament, wednesday, bakr, hostage, robinson
1 (16) NEGOTIATIONS TO SETTLE L; NEGOTIATIONS TO SETTLE L; WEST AFRICAN FORCE S
 rebel, african, taylor, west, liberia, troop, group, liberian, leader, officia
2 (28) DEATH TOLL EXCEEDS 500 I; DE KLERK, MANDELA HOLD U; COMPETING FACTIONS T
 south, police, african, black, mandela, africa, congress, anc, political, gove
3 (1) SHIFT IN U.S. COMPUTER S;
 security, agency, computer, technology, national, center, communication, milit
4 (51) SECURITY COUNCIL REACHES; @SECURITY COUNCIL REACHE; NEW U.S. POLICY IS W
 government, year, state, official, army, country, group, guerrilla, war, natio
5 (7) CLASHES BETWEEN RIVAL SH; MUSLIM FACTIONS BATTLE I; BOMBINGS IN SOUTHERN
 lebanon, muslim, christian, al, party, kill, god, lebanese, aoun, beirut, amal
6 (55) BHUTTO GOVERNMENT DISMIS; MS. BHUTTO CALLS HER OUS; MS. BHUTTO CALLS HER
 government, minister, party, political, military, prime, pakistan, president,
7 (13) SHEVARDNADZE TO VISIT TO; 45 YEARS AFTER WAR'S END; JAPAN'S ROLE IN WORL
 japan, soviet, war, korean, japanese, year, tokyo, government, south, korea, w
real time  11140 msec
```

Figure 4: Third Scatter

```
> (print-titles (nth 1 third))
 3720 REBEL LEADER SEIZES ABOUT A DOZEN FOREIGNERS
 4804 WEST AFRICAN FORCE SENT TO LIBERIA AS TALKS REMAIN DEADLOCKED
 4778 WAR THREATENS TO WIDEN AS NEIGHBORING COUNTRIES TAKE SIDES
 3719 REBEL LEADER AGREES TO HOLD CEASE-FIRE TALKS
 3409 OUSTER OF LIBERIAN PRESIDENT NOW SEEMS INEVITABLE
 3114 NEGOTIATIONS TO SETTLE LIBERIAN WAR END IN FAILURE
 3113 NEGOTIATIONS TO SETTLE LIBERIAN WAR END IN FAILURE
 2785 LIBERIANS IN U.S. CRITICAL OF ADMINISTRATION POLICY
 2784 LIBERIANS IN U.S. CRITICAL OF ADMINISTRATION POLICY
 2783 LIBERIAN REBEL LEADER CHARLES TAYLOR HURT EN ROUTE TO CEASE-FIRE
 2782 LIBERIA LEADER, REJECTING TRUCE OFFER, WON'T QUIT
 1801 FIVE WEST AFRICAN NATIONS MOVING TROOPS TOWARD LIBERIA
 1685 FACES OF DEATH IN LIBERIA
 1684 FACES OF DEATH IN LIBERIA
  248 OUSTER OF LIBERIAN PRESIDENT NOW SEEMS INEVITABLE
```

Figure 5: Titles of articles in topic 1 from Figure 4

B Group Average Clustering

Here we present a quadratic time greedy global agglomerative clustering algorithm which has given good results in our implementation. This is similar to the algorithm presented in [3].

Let Γ be a document group. The average similarity between any two documents in Γ is defined to be

$$S(\Gamma) = \frac{1}{|\Gamma|(|\Gamma|-1)} \sum_{\alpha\in\Gamma} \sum_{\beta\neq\alpha} s(\alpha,\beta).$$

Let G be a set of disjoint document groups. The basic iteration of group average agglomerative clustering finds the two different clusters Γ' and Δ' which maximize $S(\Gamma \cup \Delta)$ over all choices from G.

A new, smaller, partition G' is then constructed by merging Γ' with Δ'.

$$G' = (G - \{\Gamma', \Delta'\}) \cup \{\Gamma' \cup \Delta'\}.$$

Initially, G is simply a set of singleton groups, one for each individual to be clustered. The iteration terminates when $|G'| = k$. Note that the output from this procedure is the final flat partition G', rather than a nested hierarchy of partitions, although the latter could be computed by recording each pairwise join as one level in a dendrogram.

If we employ the cosine similarity measure, the inner maximization can be significantly accelerated. Recall that $\hat{p}(\Gamma)$ is the unnormalized sum profile associated with Γ. Then the average pairwise similarity, $S(\Gamma)$, is simply related to the inner product, $\langle \hat{p}(\Gamma), \hat{p}(\Gamma)\rangle$. That is, since

$$\begin{aligned}
\langle \hat{p}(\Gamma), \hat{p}(\Gamma)\rangle &= \sum_{\alpha\in\Gamma}\sum_{\beta\in\Gamma} \langle p(\alpha), p(\beta)\rangle \\
&= |\Gamma|(|\Gamma|-1)S(\Gamma) + \sum_{\alpha\in\Gamma}\langle p(\alpha), p(\alpha)\rangle \\
&= |\Gamma|(|\Gamma|-1)S(\Gamma) + |\Gamma|,
\end{aligned}$$

$$S(\Gamma) = \frac{\langle \hat{p}(\Gamma), \hat{p}(\Gamma)\rangle - |\Gamma|}{|\Gamma|(|\Gamma|-1)}.$$

Similarly, for the union of two disjoint groups, $\Lambda = \Gamma \cup \Delta$

$$S(\Lambda) = \frac{\langle \hat{p}(\Lambda), \hat{p}(\Lambda)\rangle - (|\Gamma| + |\Delta|)}{(|\Gamma| + |\Delta|)((|\Gamma| + |\Delta|) - 1)}$$

where

$$\begin{aligned}
\langle \hat{p}(\Lambda), \hat{p}(\Lambda)\rangle &= \langle \hat{p}(\Gamma), \hat{p}(\Gamma)\rangle + \\
&\quad 2\langle \hat{p}(\Gamma), \hat{p}(\Delta)\rangle + \langle \hat{p}(\Delta), \hat{p}(\Delta)\rangle
\end{aligned}$$

Therefore, if for every $\Gamma \in G$, $S(\Gamma)$ and $\hat{p}(\Gamma)$ are known, the pairwise merge that will produce the least decrease in average similarity can be cheaply updated each time a merge is performed. Further, suppose for every $\Gamma \in G$ the Δ were known such that

$$S(\Gamma \cap \Delta) = \max_{\Delta\neq\Gamma} S(\Gamma \cap \Delta),$$

then finding the best pair would simply involve scanning the $|G|$ candidates. Updating these quantities with each iteration is straightforward, since only those involving Γ' and Δ' need be recomputed.

Using techniques such as these, it can be seen that the average time complexity for truncated group average agglomerative clustering is $O(n^2)$ where n is equal to the number of individuals to be clustered.

References

[1] Chris Buckley and Alan F. Lewit. Optimizations of inverted vector searches. In *Proceedings of the Eighth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 97–110, 1985.

[2] W.B. Croft. Clustering large files of documents using the single-link method. *Journal of the American Society for Information Science*, 28:341–344, 1977.

[3] A. El-Hamdouchi and P. Willett. Hierarchical document clustering using Ward's method. In *Proceedings of the Ninth International Conference on Research and Development in Information Retrieval*, pages 149–156, 1986.

[4] A. Griffiths, H.C. Luckhurst, and P. Willett. Using inter-document similarity information in document retrieval systems. *Journal of the American Society for Information Science*, 37:3–11, 1986.

[5] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Pretice Hall, Engelwood Cliffs, N.J. 07632, 1988.

[6] N. Jardine and C.J. van Rijsbergen. The use of hierarchical clustering in information retrieval. *Information Storage and Retrieval*, 7:217–240, 1971.

[7] J. O. Pedersen, D. R. Cutting, and J. W. Tukey. Snippet search: a single phrase approach to text access. In *Proceedings of the 1991 Joint Statistical Meetings*. American Statistical Association, 1991. Also available as Xerox PARC technical report SSL-91-08.

[8] G. Salton. *The SMART Retrieval System*. Prentice-Hall, Englewood Cliffs, N.J., 1971.

[9] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[10] R. Sibson. SLINK: an optimally efficient algorithm for the single link cluster method. *Computer Journal*, 16:30–34, 1973.

[11] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, London, second edition, 1979.

[12] C.J. van Rijsbergen and W.B. Croft. Document clustering: An evaluation of some experiments with the Cranfield 1400 collection. *Information Processing & Management*, 11:171–182, 1975.

[13] P. Willett. Document clustering using an inverted file approach. *Journal of Information Science*, 2:223–231, 1980.

[14] P. Willett. A fast procedure for the calculation of similarity coefficients in automatic classification. *Information Processing & Management*, 17:53–60, 1981.

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