Ephemeral Document Clustering for Web Applications

Yoëlle S. Maarek[*], Ronald Fagin[†], Israel Z. Ben-Shaul[‡], Dan Pelleg[§]

Abstract

We revisit document clustering in the context of the Web. Specifically, we investigate on-line ephemeral clustering, whereby the input document set is generated dynamically, typically by search results, and the output clustering hierarchy has a short life span, and is used for interactive browsing purposes. Ephemeral clustering for interactive use introduces several new challenges. It requires an efficient algorithm, since clustering is performed on-line. It also requires high precision, because users who are not domain experts are less tolerant to errors, and because the resulting hierarchy is fully automatically generated, as opposed to off-line clustering in which the hierarchy is often manually modified. Finally, interactive clustering requires a presentation layer that enables users to effectively browse the hierarchy, including visualization techniques and automatic annotations of the hierarchy. We present new concepts, techniques and algorithms that tailor clustering to these requirements. More specifically, we improve precision by feeding the clustering algorithm with more precise profiles using "lexical affinities" as indexing units. We give an optimal complete-link Hierarchical Agglomerative Clustering algorithm, with $O(n^2)$ time complexity. Finally, we provide a visual yet light Java-based implementation of the presentation layer. We demonstrate the utility of ephemeral clustering through a search assistance utility that embodies these ideas.

keywords: information retrieval, hierarchical clustering, document clustering

IBM Research Report RJ 10186, April 2000.

[*]IBM Research in Haifa, MATAM, Haifa 31905, ISRAEL. Email: yoelle@il.ibm.com

[†]IBM Almaden Research Center, 650 Harry Road, San Jose, California, USA. Email: fagin@almaden.ibm.com

[‡]Department of Electrical Engineering, Technion, Israel Institute of Technology, Haifa 32000, ISRAEL. Email: issy@ee.technion.ac.il

[§]Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, USA. Email: dpelleg@cs.cmu.edu

1 Introduction

The field of information retrieval (IR) investigates methods and techniques for processing unstructured data such as textual data. In general, these techniques revolve around the two basic discovery paradigms, searching and browsing [Bowman et al. 1994]. Searching is performed when the user knows precisely what to look for, and browsing is performed when the user does not know exactly what to look for, e.g., when she is not familiar enough with the contents of the collection.

Browsing and searching complement each other, and are thus most effective when used in tandem. That is, browsing can be used to discover topics to search for, and search results can be organized for a more refined browsing session. The combination of searching and browsing can be also used to disambiguate otherwise ambiguous search terms, by narrowing the search to a specific category within the browsing hierarchy. Virtually all search engines combine to some extent both paradigms for "discovering" the Web. For example, Yahoo provides browsable categories in addition to global search, and it organizes the search results according to these categories. Similarly, both paradigms have been combined for facilitating "discovery" within individual Web sites (e.g., WebCutter [Maarek et al. 1997]).

An important IR technique for supporting the browsing activity is *document clustering.* The clustering process accepts a set of documents, and groups them based on their similarity. It is distinct from classification, in which an *a priori* taxonomy of categories is available beforehand, and where the documents are placed in their proper category. Clustering, in contrast, is a bottom-up process, with the categories being part of the (discovered) output, rather than part of the input. When the document collection is large and no classification scheme is available, automatic clustering is crucial for constructing browsable collections. Another advantage of clustering is that it can improve recall by returning clusters with documents that are relevant to a user query even though some of these documents may have no common terms with the original query. The advent of the Web has turned IR into a key enabling technology. Global search engines serve as the de facto Internet portals, local search engines are embedded in numerous individual Web sites, and browsing is the most common activity on the Web, due to the hyperlinked structure that provides access to more information in less space. In the context of the Web, application of clustering techniques is particularly important in two cases. First, since the document set is often very large and heterogeneous, automatic off-line grouping is important for identifying the categories, before associating documents with their proper category. In this case, clustering may be used to assist in the generation of search ontologies, and is intended to serve multiple browsing sessions. We term this kind *persistent* clustering (to clearly distinguish it from *ephemeral* clustering); it is typically used for organizing repositories of search engines. Persistent clustering is considered the "default" clustering technique, since as stated by Salton, "in normal circumstances, the cluster structure is generated only once, and cluster maintenance can be carried out at relatively infrequent intervals" [Salton 1983, p. 328]. Persistent clustering has been studied fairly extensively, and is mostly outside the scope of this work (an interesting example of large scale persistent clustering is the "fractionation" stage of the Scatter/Gather system [Cutting et al. 1992, Cutting et al. 1993]). A small-scale example on Web data is the Bookmark Organizer system [Maarek and Ben-Shaul 1996]. The second opportunity for clustering occurs when the document set is generated dynamically, by aggregating clusters of documents such as in the "gather" stage of Scatter/Gather [Cutting et al. 1992], or more typically by retrieving search results. In both cases, the output clustering is immediately used for interactive browsing purposes, in a browsing session. We denote this kind *ephemeral* clustering. Ephemeral clustering has been less studied, and is the focus of this paper.

The most common application of ephemeral clustering is for organizing and browsing search results (as suggested for traditional IR systems in [Maarek and Wecker 1994], [Hearst and Perdersen 1996] for instance), or for Web services in [Zamir and Etzioni 1999]. Organizing results is essential if one wants to see more information than the typical 10 best candidates returned by regular Web search services, without wasting time in scanning irrelevant candidates. Once the needed candidates are identified, however, the clustering that was formed especially to support the specific query can in general be discarded (although caching clusters for frequent queries may be a viable option).

The application of clustering for interactive use on dynamic collections introduces new challenges to clustering technology. First, ephemeral clustering needs to be highly *precise*, due to the user population. Whereas users of traditional clustering tools are typically professionals (e.g., librarians, lawyers, or physicians) skilled at expressing queries and experts in their respective domains, users of ephemeral clustering on the Web are less expert, and thus less tolerant to errors. Another important reason for high precision is that the resulting hierarchy that is seen by the user is generated fully automatically. This is in contrast with off-line clustering, in which the hierarchy is often manually improved (before serving multiple future browsing sessions). Second, ephemeral clustering must be *efficient*, since it is performed on-line (as opposed to the more conventional off-line clustering that is performed in digital libraries). Third, the use of the generated clustering hierarchy for interactive browsing requires a presentation layer for clustering, including a visual Web-based interface that requires a minimal learning curve to support casual users, as well as minimal interaction in terms of user feedback. Indeed, while user feedback – such as relevance feedback on search results or the "gather" stage on clustering output in Scatter/Gather – clearly improves the quality of any discovery tasks, it is still used only by a limited class of users (mostly early adopters, or digital librarians).

This paper presents a collection of new concepts, techniques, and algorithms for ephemeral clustering that addresses these requirements and demonstrates their utility through an actual tool that embodies these ideas. The rest of this paper is organized as follows. Section 2 gives an overview of traditional document clustering and existing Web-based clustering techniques. Section 3 describes our contribution in improving the effectiveness of ephemeral clustering through precise indexing techniques. Section 4 describes our contribution in improving the efficiency of ephemeral clustering through an $O(n^2)$ hierarchical complete-link algorithm. Section 5 discusses the presentation layer and describes a search assistance tool. Finally, Section 6 summarizes the contributions of this paper and points to future directions.

2 Clustering Overview and Related Work

Cluster analysis has been of long-standing interest in statistics, numerical analysis, machine learning (where it is commonly called unsupervised learning) and other fields. Some trace it back to the work of Adanson as early as 1757 for classifying botanic species [Adanson 1757]. Current cluster analysis offers a wide range of techniques for identifying underlying structures in large sets of objects and for revealing relationships between objects or classes of objects. A *cluster* intuitively corresponds to a group of objects whose members are more similar to each other than to the members of other clusters. Typically, the goal of cluster analysis is to determine a *clustering*, that is, a set of clusters, such that intra-cluster similarity is high and inter-cluster similarity is low. Clustering methods are usually classified according to two aspects: the generated structure, which could be hierarchical, flat, or overlapping; and the technique used to implement the structure, including divisive (start from a set of objects and split it into subsets, possibly overlapping) and agglomerative (start from individual objects, i.e., singletons, and merge them into clusters). Various clustering methods are used in various fields of applications. Flat non-overlapping clustering is popular in pattern recognition applications [Michalski and Stepp 1983] such as identifying shapes as disjoint clusters of pixels in an image. In contrast, overlapping methods allow objects to be members of more than one cluster. In the context of document clustering, the overlapping corresponds to the useful notion that the same document may belong to several unrelated topics. An advantage of non-hierarchical methods is performance. It is in general faster to generate a flat list than a hierarchy. Thus, for on-line clustering, flat methods (overlapping or partitioning) have often been preferred mostly because they require only linear (under certain constraints) time complexity [Cutting et al. 1992], [Zamir and Etzioni 1999] (or even constant time via some simplifications, e.g., [Cutting et al. 1993].)

The overriding argument in favor of hierarchical clustering, however, is that it is much more effective for browsing, because it enables the user to do a logarithmic-time traversal of the tree from general to more specific topics, as opposed to the linear-time traversal of non-hierarchical methods. This is particularly the case if internal (generated) nodes can reveal information about their contained sub-hierarchies (we will show such support in Section 5). It also presents the advantage over many (but not all) divisive methods of not requiring the *a priori* definition of the ideal number of clusters. Unlike those methods, hierarchical clustering does not impose a predefined structure over a set of objects. As we will see in the following, the clustering output can then be controlled by the degree of cohesion of each cluster, rather than by an arbitrary number of clusters or elements per cluster.

Our choice of hierarchical clustering introduces the challenge of good performance so as to enable ephemeral clustering. Indeed, classical Hierarchical Agglomerative Clustering methods that have been used for document clustering in the IR community have a time complexity of $O(n^3)$ [Voorhees 1986a, Frakes and Baeza-Yates, 1992] and $O(n^2 \log n)$ [Salton 1983], or at best $O(n^2)$ [El-Hamdouchi and Willett 86, Murtagh 1984] (where n is the number of documents to be clustered) as discussed later. Although the typical input document set for ephemeral clustering is much smaller than for off-line clustering, achieving optimal performance is highly important. We show in Section 4 how the performance issue is addressed.

In order to keep the discussion self-contained, we introduce basic terminology and concepts in hierarchical clustering. The uninitiated reader may refer to [Everitt 1980] for a detailed discussion. We begin by considering a notion of *similarity* between documents. The similarity, or rather the dissimilarity, between documents is typically measured by a *dissimilarity index*, which can be thought of as the "distance" between two documents (except that a dissimilarity index need not satisfy the triangle inequality). More formally, a function δ is a *dissimilarity index* over a set Ω of objects if it satisfies the following properties:

1. $\delta(o, o) = 0$ for each object o in Ω, and
2. $\delta(o, o') = \delta(o', o)$ for each pair (o, o') in Ω^2.

Note that if a measure of similarity – rather than of dissimilarity – is available over a set of objects, then it is straightforward to derive from it a dissimilarity index. In IR, for instance, it is common to define the dissimilarity index over a set of documents as 1 minus a given similarity measure. The similarity measure is a number between 0 and 1, where the similarity measure of a document to itself is 1.

In the vector space model [Salton 1983], each possible indexing unit is associated with one dimension and each document is represented by a profile vector. The most common similarity measure between two documents is the cosine coefficient between their associated profile vectors after normalization, which takes into account the size of the documents (which equals the scalar product, when the vectors are normalized to be of length 1).

An indexing unit can be a single term (possibly represented by a canonical form such as its morphological root, lemma or stem), or it may take more complex forms, such as phrases, syntactic constructs [Fagan 1989] or lexical constructs (as shown later in Section 3). For example, if the indexing units are single words, then each word represents an axis in a high-dimensional vector space, where the dimension is equal to the number of words in the collection. Thus, a document vector has coordinate 0 on dimensions associated with words that do not appear in the document, and a positive value, which represents a "score", on dimensions that are associated with words that the document contains. The score can be the frequency of the word in the document, possibly normalized. More sophisticated scores reflect the degree to which an indexing unit is representative of the document, within the context of a given collection (see [Frakes and Baeza-Yates, 1992] for a survey of scoring methods).

The most common hierachical clustering technique, that has been used extensively in IR, is Hierarchical Agglomerative Clustering (HAC). It requires not only the definition of a similarity measure between individual documents, but also the definition of a similarity between clusters of documents. Three common methods for defining similarity between clusters are the single-link method, the complete-link method, and Ward's method. In the single-link method, the similarity between two clusters C_1, C_2 is the *maximum* of the similarity between a pair d_1, d_2 where $d_1 \in C_1$ and $d_2 \in C_2$. Thus, two clusters are similar if *some* pair of members are similar. In the complete-link method, [Voorhees 1986a], the similarity between two clusters C_1, C_2 is the *minimum* of the similarity between a pair d_1, d_2 where $d_1 \in C_1$ and $d_2 \in C_2$. Thus, two clusters are similar if *every* pair of members are similar. In Ward's method [Ward 1963], the pair of clusters that are considered to be closest together among all clusters (and hence are merged in a step of a hierarchical clustering algorithm) is the pair whose merger minimizes a certain sum of squares error based on distances from centroids of the clusters. The complete-link method is usually favored in IR applications, because it tends to produce smaller and more tightly-linked clusters, compared with other methods [Voorhees 1986a]. Intuitively, in a cluster formed by the complete-link method, every member is similar to every other member. However, unlike single link and Ward's[1] methods that have known implementations in time $O(n^2)$, the best complete-link document clustering algorithm as far as we know is $O(n^2 \log n)$. We propose here an optimal $O(n^2)$ complete-link algorithm that uses certain simplifying assumptions specific to document clustering.

The reader should consult [Everitt 1980] for an extensive survey of HAC approaches and [Willet 1988], [Rasmussen 1992] for their application to IR. The HAC approach can be described as follows. Given a set of documents:

- Start with a set of singleton clusters, each containing one document, and each initially unmarked.
- Repeat the following steps iteratively until there is only one cluster left unmarked.
  - Identify the two most similar unmarked clusters, and mark them.
  - Form a (new, unmarked) parent for these clusters by merging them together into a single cluster.

Since after each iteration there is one less unmarked cluster, this process always terminates. The HAC approach iteratively builds a sequence of partitions or *level clusterings* of the original set Ω of documents. The level clusterings form coarser partitions beginning with the level clustering formed by the set $\{\{o_1\}, \{o_2\}, \ldots, \{o_n\}\}$ of singletons, and ending up with the coarsest partition $\{\Omega\}$ of Ω. Each cluster in a given level clustering can be seen as a node in a hierarchy. When the cluster is a result of a merge, its direct descendants are its two subclusters in the previous level. The final output of this clustering process is a particular form of tree called a *dendogram*, in which every internal node has exactly two children nodes. A sample dendogram is depicted in Figure 1, with objects $o_1, o_2, \ldots, o_9$ on the x-axis, and the level of similarity at which the cluster was formed on the y-axis (which is by definition between 0 and 1).

Dendograms were traditionally used for improving retrieval efficiency. The idea was to compare queries only to cluster centroids rather than to all documents, thereby reducing significantly the number of necessary comparisons. Efficiency was improved but at the cost of retrieval effectiveness [Salton 1971]. Later, dendograms were used "as is" based on the assumption that "associations between documents convey

[1] See [El-Hamdouchi and Willett 86] for the application of Ward's method to document clustering.

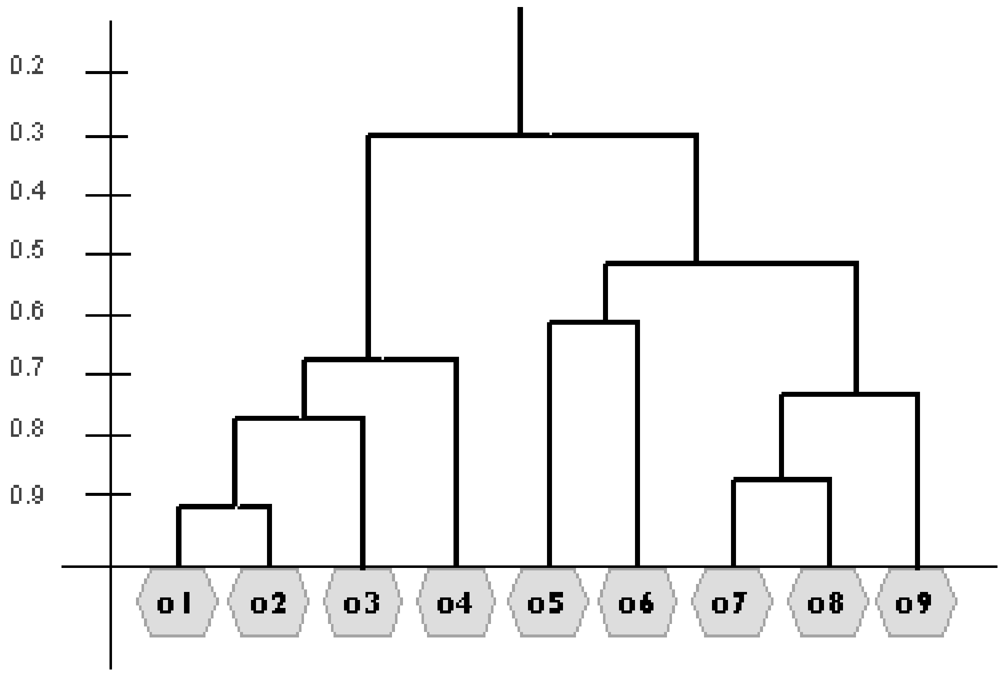


Figure 1: An example of dendogram

information about the relevance of documents to requests" [Jardine and van Rijsbergen 1971]. It is this latter use of dendograms that is of interest in this work.

3 Improving Effectiveness of Ephemeral Clustering

Recall that a key requirement for ephemeral clustering is precision. Unlike persistent clustering, whereby quality control can be applied to the result (e.g., by manually modifying the hierarchy), in ephemeral clustering the resulting hierarchy is to be used as is (i.e., it is fully automated). Thus, as a general guideline, precision is preferred over recall. In practice, this means that when in doubt, a document is not classified, in order to avoid the possibility of associating it with a wrong cluster. Instead, the document is left as an outlier.

As for improving the precision of the clustering process, recall that the HAC algorithm requires a similarity measure between documents, by considering as input a matrix of pairwise similarities on the set of documents to be clustered. Instead of attempting to improve the effectiveness of HAC itself, we propose to improve the quality of its input, i.e., improve the quality of the profile vectors on which to apply the similarity measure. This is achieved through the following scheme. Instead of the typical use of single words as indexing units, our indexing unit consists of a pair of words that are linked by a *lexical affinity* (LA). An LA between two units of language stands for a correlation of their common appearance [Saussure 1949]. It has been described elsewhere how LAs can be extracted from text at a low cost via a $+/-5$ word sliding window technique[2]. One key advantage of LAs over phrases is that they represent more flexible constructs, that link words not necessarily adjacent to each other (for more information on LAs see [Maarek and Smadja 1989]).

In the LA-based IR system described in [Maarek et al. 1991], we used profiles mixing both LAs and single words, partly because many users issue single-word queries, and partly because precision cannot be preferred too significantly over recall in an arbitrary search application. Here, however, precision is the key criterion, and each document is at least a few sentences long and therefore, we suggest to use exclusively LAs as indexing units. We will justify this decision via experimental results later on.

Before doing so, let us illustrate via some examples how LA-based profiles can improve the precision of the pairwise similarity scores and therefore of the clustering output. Let us consider for instance, two Web pages returned by the Google (`http://www.google.com`) search service among the results to the query ``` ``merced´´ ```. One example is a page from the "Merced County" Web site.[3] Table 1 shows the single-word and LA profiles of this page. One can immediately see in that example that LAs are more informative than single words, and provide de facto disambiguation. Thus, the very first LA "county*merced"[4] refers to the key concept of the page. The three following LAs stand for a ternary[5] lexical affinity, namely, "yosemite national park". By contrast, in the single word list, while the first item is indeed the key concept "merced", the following items are not as informative. For instance, one could wonder what the relation of the 4th item "hour", is to this page. The 5th LA "drive*hour" carries the answer: the page probably specifies commute distances in "hour drive" units.

Single words	**Lexical Affinities**
0.37 merced	0.20 county*merced
0.29 yosemite	0.13 national*park
0.12 county	0.13 national*yosemite
0.12 hour	0.13 park*yosemite
0.08 populate	0.08 drive*hour
...	...

Table 1: Single-word vs. LA-based profiles for the Merced County Web page

[2]Based on Martin's results that, in English, 98% of LAs appear in a window of 5 words, [Martin et al. 1983].

[3]Specifically at URL `http://www.co.merced.ca.us/About_us/index.html`

[4]LAs are represented here as pairs of lexically ordered words; the order of appearance of the words is less crucial than in phrases.

[5]Since any ternary LA can be represented as a set of binary LAs, it is in most cases worthwhile from a computational viewpoint to compute only binary LAs.

Let us consider now the profile of another result page for the "merced" query, namely a news story entitled "Intel and HP, Merced chip's reluctant partners"[6]. This page, while totally unrelated to the Merced county page previously mentioned, is relevant to another sense of "merced". Interestingly enough, since the word "merced" is very highly ranked in both pages (see Table 2), the pairwise similarity score required by the HAC algorithm as explained in the previous section would probably have a non-negligible value for these two documents even if they match on one single indexing unit. Consequently they could be merged into a "wrong" cluster. In contrast, if using exclusively LA-based profiles, these two documents share no indexing unit whatsoever, and therefore would have a null similarity score, and would not be merged into a cluster (except the topmost cluster consisting of all documents).

Keywords list	**Lexical Affinities**
0.39 merced	0.20 chip*intel
0.19 intel	0.13 hp*intel
0.15 hp	0.08 chip*merced
0.13 risk	0.08 intel*merced
0.13 ia	0.08 backward*compatible
...	...

Table 2: Single-word vs. LA-based profiles for the Intel Chip news story

3.1 Evaluation

In order to assess the quality of a hierarchy, there is a need to examine how close it is to the "ground truth". That is, how helpful is the knowledge that a specific document belongs in cluster X of the hierarchy? Naturally, we would want this kind of fact to have a strong implication about the document belonging in a "true" class Y. In general, we would want the clusters generated by our hierarchy to be as homogeneous as possible, with respect to the labeling of the contained documents, as induced by the "ground truth" classification. A natural measure of homogeneity is information-theoretic entropy [Shannon]. Intuitively, if the hierarchy producer Alice wants to transmit the classification of documents in a given cluster to Bob, then the entropy is the expected number of bits they would exchange over a serial communication channel before the classification is made known to Bob. One major drawback of this approach is that it does not account for the cost of transmitting the hierarchy itself. In particular, hierarchies that contain only singleton clusters are optimal under this measure. Each singleton cluster is highly informative (no more bits are needed to be sent, since the class is known perfectly). The fact remains, however, that this kind of hierarchy is practically useless.

Recently, Dom [Dom 2000] has refined the entropy measure to include the cost of the model (the

[6]From the page `http://www.zdnet.co.uknews/1998/45/ns-6066.html`

clustering). His formula penalizes both highly heterogeneous clusters and a large number of clusters. Hence, on one extreme of his "clustering quality" measure, we have a single cluster, which is just as informative as the true classification, and very cheap to transmit, while on the other end we have the all-singleton case, which is highly informative but very expensive to transmit. The optimum is usually achieved somewhere in between. Again, this measure is the expected number of bits to transmit the classification to another party (this time including the cost of transmitting the model).

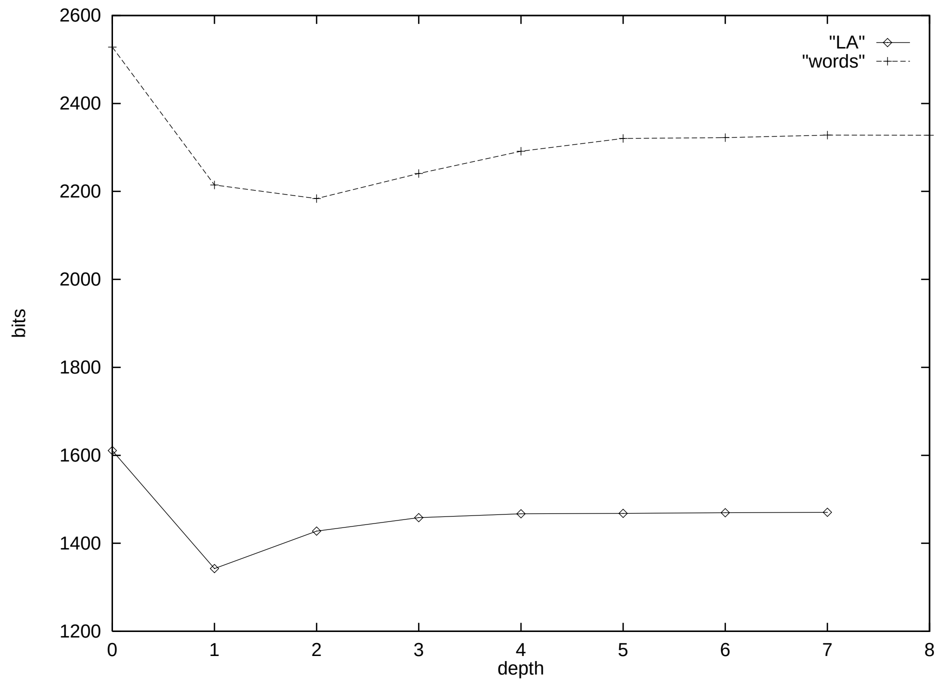


Figure 2: A Clustering Quality Measure for the "cleaned" experiment

Since the "clustering quality" measure works only for partitions and not for hierarchies in general, we "flattened" the results of our hierarchies in order to compute the clustering quality. Define a partition for depth d by taking each node (or more exactly, the documents within that node) to be a member of the partition. We then evaluate our hierarchy based on the evaluations of the partitions at each depth.

For evaluation, we used the Reuters-21578 dataset [Lewis 1999]. Our "ground truth" classification was the manually-labeled topics of articles in the dataset. We extracted 43 sets of documents. Each set contained 10 articles, all with the same (unique) topic[7]. We then ran ephemeral clustering on the entire 430 document set (not using the manual labels). For a resulting hierarchy we plotted the clustering quality measure as a function of the depth, as explained above. Another experiment included a "cleaning" stage, where direct children of the root, that is, outliers, were removed from the resulting hierarchy before the quality measure was applied. Both experiments were performed twice, one using just LAs as the document features, and the other using just single words as the features. The results for the "cleaned" experiments are shown in Figure 2. The results for the "raw" experiments are shown in Figure 3.

One obvious observation from both plots is the trade-off between hierarchy complexity and classification accuracy. We see that the number of bits needed is large at the extremes (corresponding to all-singletons and single-class clusterings), and is minimized at depth 1 or 2. This might suggest that clearer visualization of the hierarchy may be obtained if the tool pruned the resulting hierarchy at one of those levels by default[8].

The results in Figure 2 clearly show that using LAs improves the quality of the hierarchy by approximately 30%, compared with the same algorithm which uses single words as document features. Although not a new discovery, this is yet another piece of evidence for the superiority of LAs. As for the results in Figure 3, they are not conclusive. In fact, it shows that single words fare better than LAs in most cases. Recall that the difference from Figure 2 is the inclusion of the direct children of the root in the resulting hierarchy. For these documents, their cohesion score was so low they did not get clustered until the very last step. Intuitively, these documents contain noise, similar to the unrelated documents that a low-quality Web search engine might produce. We can assume that by omitting them we lose no information. Furthermore, our results show that doing so improves the informativeness of the hierarchy (the expected number of bits generally goes down between the "raw" and "clean" cases). These results also show that LA-based techniques are better suited to pick up the underlying classification once this noise (represented by outliers) is removed.

4 Improving Efficiency of Ephemeral Clustering

In [Salton 1989], Salton suggests a lower bound on a clustering algorithm that uses a HAC approach with a complete-link method:

> "For n items, there will be $n(n-1)/2$ pairwise similarities, and any sorting operation necessary to arrange the similarity pairs in decreasing order of similarity will require of the order of $n^2 \log n^2$

[7] Articles in the Reuters dataset may have multiple labels. We consider here only articles with exactly one label.

[8] This is under the assumption that communication efficiency directly translates to human readability.

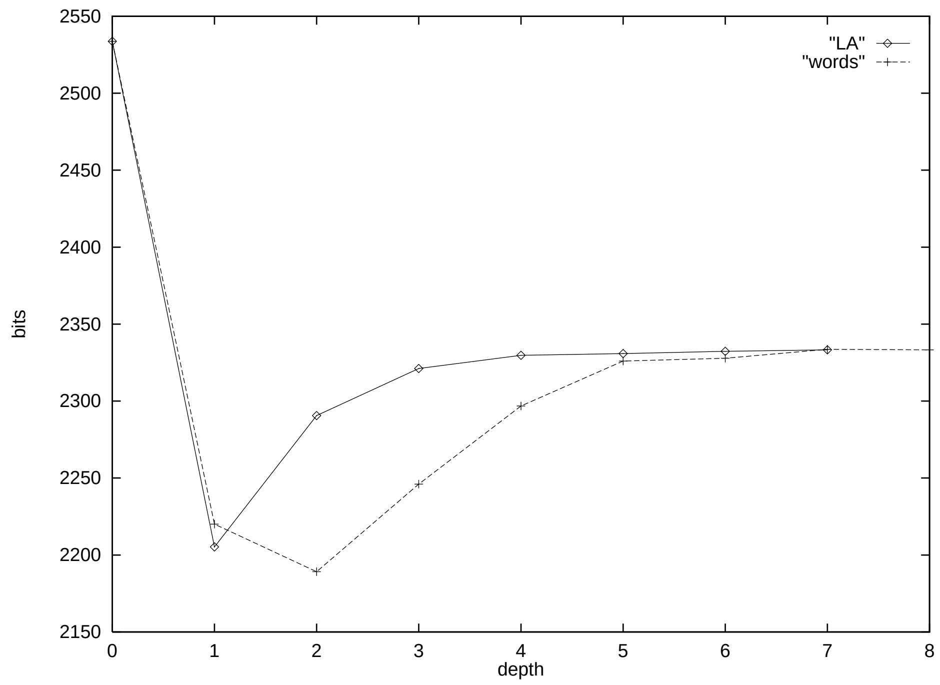


Figure 3: A Clustering Quality Measure for the “raw” experiment

operations."

Note that, as observed by Voorhees [Voorhees 1986b], fewer than the theoretical $n(n-1)/2$ comparisons are necessary. Indeed, one can take advantage of the fact that each document has a non-zero similarity with relatively few other documents. This phenomenon is even more evident when only the best indexing units are kept in each profile, and when using lexical affinities rather than single words as indexing units.

We propose to reduce the $O(n^2 \log n^2) = O(n^2 \log n)$ upper bound that seems to be inherent in the sorting step based on the following observation: if the range of values (the interval [0,1]) and the granularity of values within the range are known and small, a simple linear bucket sorting is possible. Fortunately, in clustering hierarchies that are intended for browsing purposes, keeping a relatively coarse granularity (typically 0.1) for the precision of the similarity measure is actually preferable over finer granularities. The reason is that a user does not need to distinguish between many different levels of similarity that are so close. He would rather know what clusters correspond to a degree of similarity of approximately 90%, 80%, etc. In fact, the binary dendogram that is generated by a clustering algorithm that is based on precise similarities is inconvenient for visualization and browsing. Therefore, when used for classification of animal species, chemical structures, etc., experts try to enhance the shape of the graph by increasing the branching factor, i.e., by merging several nodes under the same parent. In a previous work [Maarek et al. 1991, Maarek and Wecker 1994], several pruning methods were invented to be applied a posteriori, in a second pass on the binary dendogram, in order to obtain a more readable tree. Our new algorithm generates directly a more compact and readable structure than the binary dendogram generated by HAC, thereby eliminating the need for an additional post processing "correction" phase. The branching factor in the resulting hierarchy is effectively determined by the level of granularity that is chosen for the similarity. Figure 4 shows an original binary dendogram (left figure), and a modified dendogram (right figure) with less levels (4 instead of 7 in our simple example), where the reduced number of levels is obtained in one pass, rather than by slicing the original dendogram as in [Maarek and Wecker 1994].

Bucket sorting requires only $O(m)$ steps, rather than $O(m \log m)$ steps, to sort m elements. In our case, where the m elements are the $n(n-1)/2$ pairwise similarities between n documents, this corresponds to only $O(n^2)$ steps. We shall take advantage of this fact to obtain an $O(n^2)$ HAC algorithm, by allowing only a linear number of additional steps, each of which require only linear time. We note that Murtagh [Murtagh 1984] has given an $O(n^2)$ algorithm for generating an output equivalent to that which a complete-link HAC algorithm would give. Interestingly enough, in Murtagh's algorithm the clusters are produced in a different time order than they would be with a traditional complete-link HAC algorithm, although the same clusters are eventually produced.

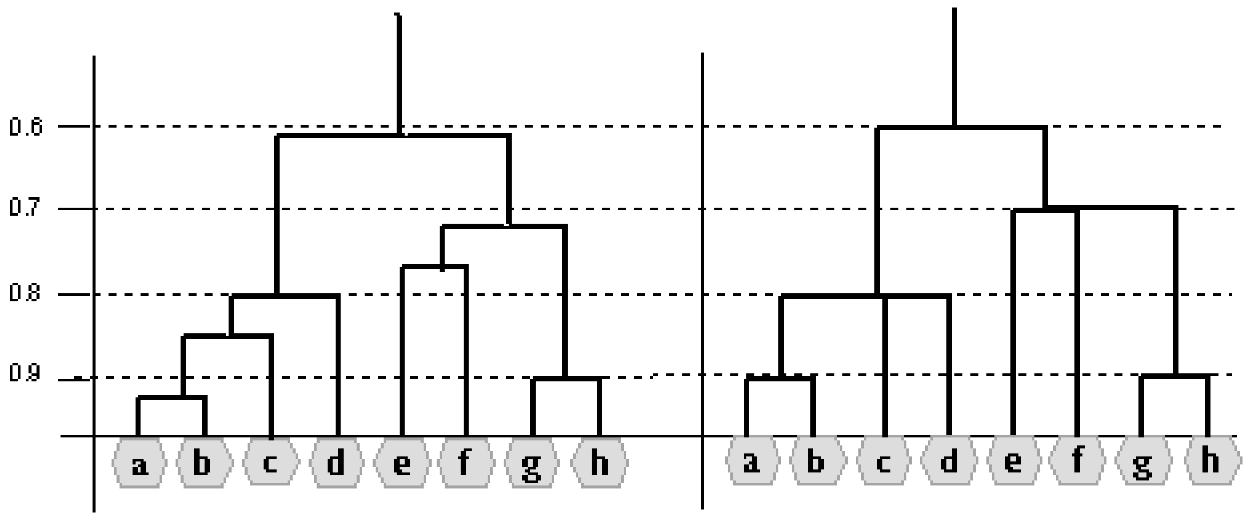


Figure 4: Binary Dendogram vs. Compacy Hierarchy Generated by the Algorithm

Data Structures

We consider clusters to be sets of documents. By rounding down if necessary, we shall assume for simplicity in description that the similarity of each pair of distinct documents is one of the ten values 0, 0.1, 0.2, ..., 0.9. By "rounding down", we mean that we replace each similarity value s by $\lfloor 10s \rfloor / 10$. Note that we are assuming for simplicity that no two distinct documents have similarity 1.0. We define the *similarity value* of a cluster c to be the minimum of the pairwise similarities of members of the cluster c. This corresponds to the fact that we are using the complete-link method. Similarly, we define the similarity value of a pair (c_1, c_2) of clusters to be the minimum of the pairwise similarities of members of the union of c_1 and c_2. Thus, the similarity value of the pair (c_1, c_2) is what the similarity value would be of a new cluster that would result by combining c_1, c_2 into a single cluster.

We maintain a matrix (the "current similarity matrix") that tells the similarity value of each pair (c_1, c_2) of unmarked clusters c_1 and c_2 (unmarked clusters are those clusters that are eligible for being combined into a new cluster). In particular, the diagonal entry corresponding to (c, c) tells the similarity value of cluster c. After each iteration of our algorithm, both dimensions of the current similarity matrix are reduced by one.

We maintain ten buckets, called 0, 0.1, ..., 0.9, which are each represented by doubly-linked lists of pairs (c_1, c_2) of distinct unmarked clusters. (Intuitively, these pairs (c_1, c_2) in the doubly-linked list are the members of the bucket.) Bucket θ corresponds to cluster pairs (c_1, c_2) whose similarity value is θ (which means that if we had not rounded down, the similarity value would have been in the interval $[\theta, \theta + 0.1)$. The "order" in the linked list within a bucket is arbitrary. The buckets are updated at each iteration. To help in this updating process, there is a matrix (the "current pointer matrix") that for each pair (c_1, c_2)

of distinct unmarked clusters, points to where in the doubly-linked list of the appropriate bucket the pair (c_1, c_2) appears. Both the current similarity matrix and the current pointer matrix are symmetric about the diagonal.

The Body of the Algorithm

1. The initial set of clusters are the singleton sets, one for each of the n documents. These are each taken to be unmarked. The current similarity matrix is initialized by taking it to be an $n \times n$ matrix that gives the similarity value of each pair of documents. Each similarity value s is rounded down to $\lfloor 10s \rfloor/10$. The buckets are initialized by placing each pair (c_1, c_2) of distinct singleton sets into the appropriate bucket. In doing so, the current pointer matrix is initialized so that for each pair (c_1, c_2) of distinct singleton sets, the (c_1, c_2) entry points to where in the doubly-linked list of the appropriate bucket the pair (c_1, c_2) appears.[9]

2. The nonempty bucket with the biggest index θ is found, along with the pair (c_1, c_2) at the "top" of the list for this bucket. There are then three cases.

   Case 1: the similarity values of (c_1, c_2), c_1, *and* c_2 *are all the same.* In this case, c_1 and c_2 are merged into a single cluster (which we call here c_1c_2, to represent the fact that it contains the members in the union of c_1 and c_2). Intuitively, the new unmarked cluster c_1c_2 is created, and the clusters c_1 and c_2 are marked and discarded. The set of children of c_1c_2 is taken to consist of those clusters that were the children of either c_1 or c_2.

   Case 2: the similarity value of (c_1, c_2) *is less than the minimum of the similarity values of* c_1 *and* c_2*.* In this case, the clusters c_1 and c_2 are marked, the new unmarked cluster c_1c_2 is created, and c_1 and c_2 are taken to be the children of c_1c_2.

   Case 3: the similarity value of (c_1, c_2) *is equal to the minimum but less than the maximum of the similarity values of* c_1 *and* c_2*.* Let us assume without loss of generality that the similarity value of c_1 is less than the similarity value of c_2. In this case, the new unmarked cluster c_1c_2 is created, the clusters c_1 and c_2 are marked, and the cluster c_1 is discarded (intuitively, the cluster c_1 is replaced by the cluster c_1c_2). The set of children of c_1c_2 is taken to consist of c_2, along with the children of c_1. Note that c_1 had at least two children, since it is easy to see that each non-singleton cluster has at least two children (c_1 is not a singleton cluster, since the similarity value of c_1 is less than 1.0).

3. The data structures are updated as follows.

   (a) The similarity value given in the new diagonal entry (c_1c_2, c_1c_2) in the current similarity matrix is taken to be the (c_1, c_2) entry in the current similarity matrix. Then the similarity value of

[9] For efficiency, we actually put either (c_1, c_2) or (c_2, c_1), but not both, into a bucket, and we take the (c_1, c_2) and (c_2, c_1) entries in the current pointer matrix to be the same.

(c_1c_2, c) (and of (c, c_1c_2)) for each unmarked cluster c other than c_1c_2, is computed by taking the minimum of the similarity values of (c_1, c), (c_2, c), and (c_1, c_2). Thus, a column and a row. each corresponding to c_1c_2, are added to the current similarity matrix. Then the columns and rows corresponding to c_1 and c_2 are removed. The net effect is to reduce each dimension of the current similarity matrix by one.

(b) As the information about the similarity value of (c_1c_2, c) is computed, the pair (c_1c_2, c) is added to the appropriate bucket b. The current pointer matrix is updated so that the (c_1c_2, c) and (c, c_1c_2) entries point to where in the doubly-linked list in the bucket b the pair (c_1c_2, c) appears. This involves adding a new column and row, each corresponding to c_1c_2. Then (c_1, c) and (c_2, c) are removed from their buckets, by using the pointer information in the current pointer matrix.[10] Finally, the columns and rows corresponding to c_1 and the columns and rows corresponding to c_2 are removed from the current pointer matrix. The net effect is to reduce each dimension of the current pointer matrix by one.

4. Stop when there is only one unmarked cluster. Note that this occurs after the iteration where the dimension of the current similarity vector and the dimensions of the current similarity matrix and the current pointer matrix are each one.

Complexity Analysis

1. The initialization takes time $O(n^2)$ to initialize the current similarity matrix, time $O(n^2)$ to initialize the buckets, and time $O(n^2)$ to initialize the current pointer matrix.
2. This step takes time $O(1)$.
3. This step takes time $O(n)$.
4. There are only $n - 1$ iterations, since at each iteration, the number of unmarked clusters decreases by one.

Thus, the initialization takes time $O(n^2)$, and then there are a linear number of iterations, each of which take a linear amount of time. So altogether, the total time is $O(n^2)$.

5 Applying Ephemeral Clustering to the Web

So far, we have described techniques for enhancing the effectiveness and efficiency of ephemeral clustering, and showed how our method enhances browsing applications by reducing the number of levels in the

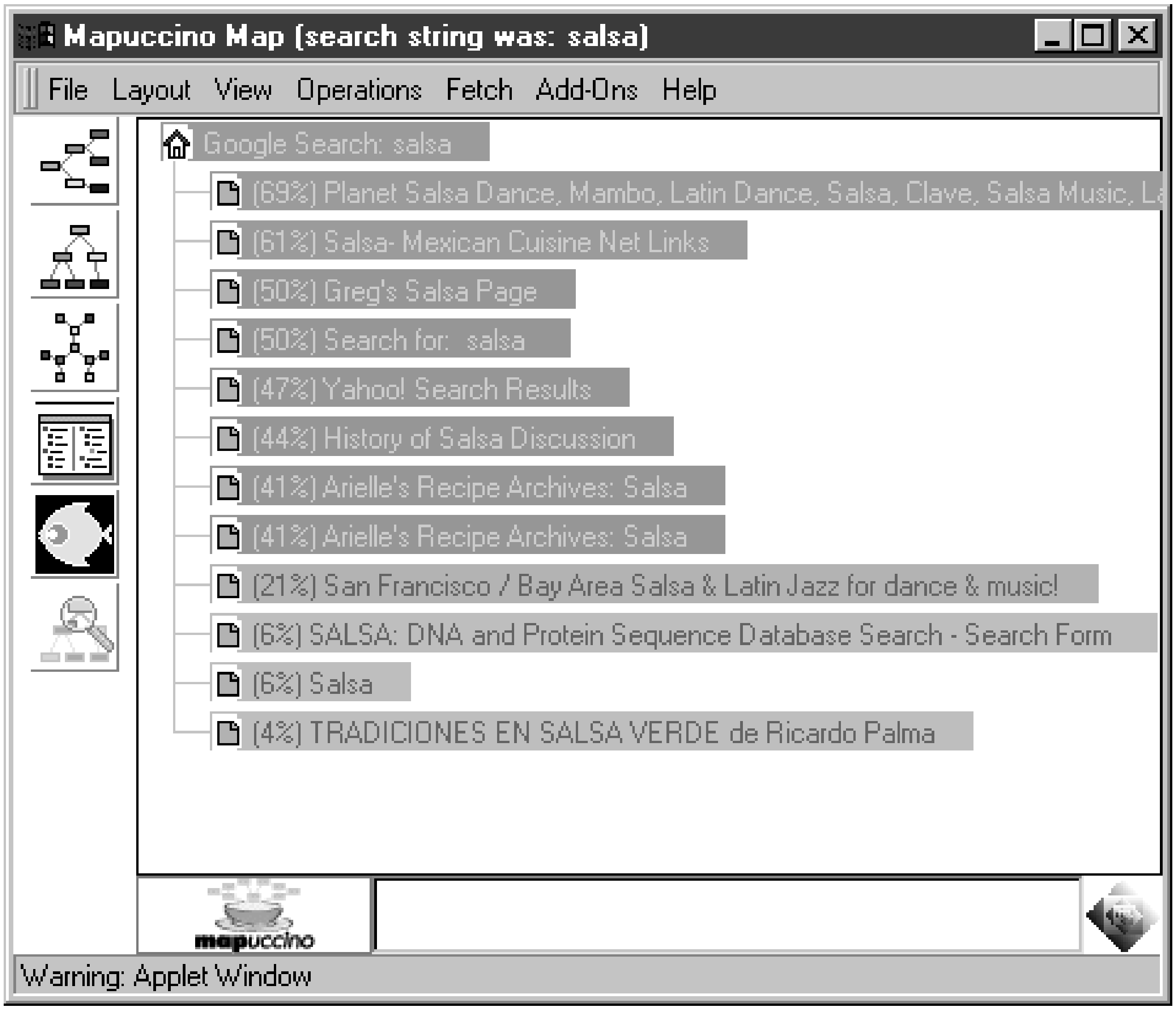
[10] Instead of (c_1, c), the bucket might have contained (c, c_1), which is removed from the bucket; see the previous footnote. A similar comment applies, of course to (c_2, c) and (c, c_2).

hierarchy. While implementing an actual tool that utilizes these techniques, however, we discovered additional issues that needed to be addressed, mostly relating to the interaction with casual end-users, which has been traditionally less of an issue for non-ephemeral clustering applications. First, a clustering technique that automatically generates user-oriented hierarchies must provide a method for (automatically) annotating internal nodes with information that conveys the contents and degree of similarity. Second, the visualization of the clustering hierarchy in general is instrumental for effective use of such tool.

The Lassi (Librarian's ASSIstant) tool addresses these issues. Lassi receives as input a set of HTML documents (typically the result of a search) and generates a clustering hierarchy as output using the LA-based indexing and the clustering algorithm described above. We have integrated Lassi features into our search parasite, Fetuccino, whose purpose is to take search results from various search services, identify links between them, and pursue directed crawling in the most relevant directions (See [Ben-Shaul et al. 1999] for more information). Fetuccino results can be laid out as a graph or simply as a traditional ranked list. Figure 5 shows how the user can invoke the clustering feature provided by Lassi from within Fetuccino.

Lassi performs the indexing and clustering of the documents on the fly and generates a clustering hierarchy as shown in Figure 6. The resulting hierarchy is displayed visually via a Java applet (derived from the original Mapuccino/Fetuccino applet) that allows the user to browse the hierarchy. For example, clicking on a leaf (document) node pops up the document contents in a separate window, while clicking on an internal (cluster) node shows information about the cluster (this is further elaborated below). Nodes that are closer to the leaves represent more cohesive clusters; the closer that nodes are to the root, the less specific they are.

We turn now to presenting Lassi's automatic labeling and its general visualization scheme. Figure 6 demonstrates some key benefits of the hierarchical clustering process. It shows the clustering of search results of the query "salsa". The results show that 3 key senses of the word "salsa" are identified (dance, music and food), and the third cluster once open reveals more specific clusters within the same domain. This shows the benefits of hierarchical clustering over flat clustering, as it provides a more organized information rather than the 6 or 7 clusters "flat" generated by Grouper on exactly the same query. Note however that the clustering process is not perfect, and some associations are still missed, mainly because of the preference given to precision. For instance, among the three outliers listed at the bottom of Figure 6, while the first one (about the DNA database) is rightly left unclassified, and the third one is understandably not classified since the page is entirely in Spanish, the second one deals with music and would have ideally been added to the second cluster. However, since it does not share enough information with that cluster (it mainly contains a long list of salsa music performers), it was not classified. While this could be considered a flaw, we believe that even if the number of outliers is large, it is still preferable to retain a high-level of confidence rather than providing noisy information.


Mapuccino Map (search string was: salsa)
File Layout View Operations Fetch Add-Ons Help

Google Search: salsa

(69%) Planet Salsa Dance, Mambo, Latin Dance, Salsa, Clave, Salsa Music, La

(61%) Salsa- Mexican Cuisine Net Links

(50%) Greg's Salsa Page

(50%) Search for: salsa

(47%) Yahoo! Search Results

(44%) History of Salsa Discussion

(41%) Arielle's Recipe Archives: Salsa

(41%) Arielle's Recipe Archives: Salsa

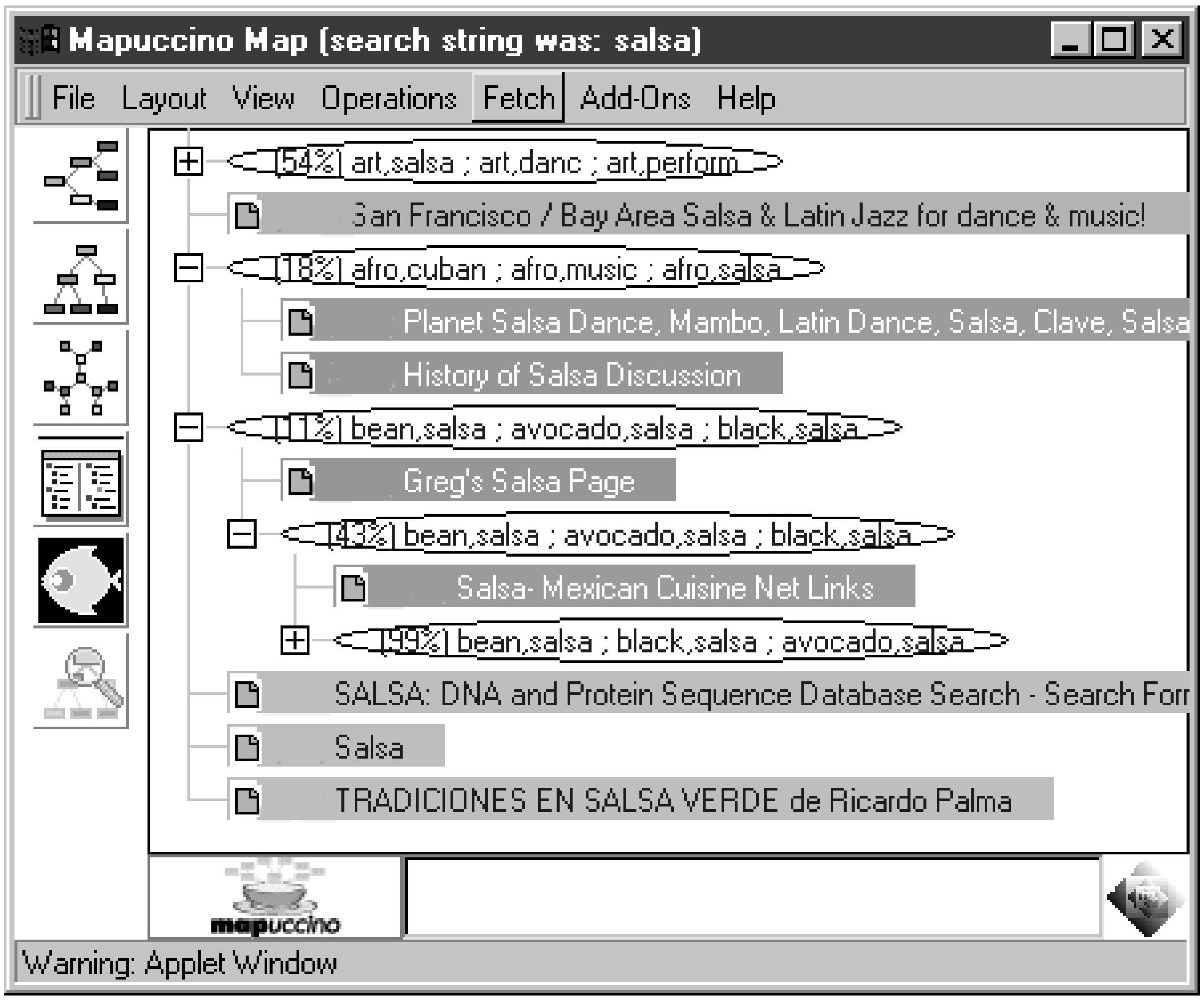
(21%) San Francisco / Bay Area Salsa & Latin Jazz for dance & music!

(6%) SALSA: DNA and Protein Sequence Database Search - Search Form

(6%) Salsa

(4%) TRADICIONES EN SALSA VERDE de Ricardo Palma

Figure 5: Invoking Lassi Clustering on Fetuccino Search Results



```
[54%] art,salsa ; art,danc ; art,perform
    San Francisco / Bay Area Salsa & Latin Jazz for dance & music!
[18%] afro,cuban ; afro,music ; afro,salsa
    Planet Salsa Dance, Mambo, Latin Dance, Salsa, Clave, Salsa
    History of Salsa Discussion
[11%] bean,salsa ; avocado,salsa ; black,salsa
    Greg's Salsa Page
    [43%] bean,salsa ; avocado,salsa ; black,salsa
        Salsa- Mexican Cuisine Net Links
        [99%] bean,salsa ; black,salsa ; avocado,salsa
SALSA: DNA and Protein Sequence Database Search - Search For
Salsa
TRADICIONES EN SALSA VERDE de Ricardo Palma
```

Figure 6: Output of Lassi clustering in "tree-control view"

5.1 Automatic Node Labeling

One drawback of clustering (compared to classification) is that clusters are defined by extension (i.e., by enumeration of their members), rather than by intention (i.e., by membership rules). Therefore, there is no direct way to name the identified clusters. This is a serious problem in our context, where users are not familiar with the data set (it can be drawn from the entire Web) and where the labels on the leaves (the documents being clustered) are not always informative (e.g., a URL). We address this problem via a simple method of labeling each cluster with a list of characterizing key concepts, termed pseudo-titles.

The general idea is to select a set of index terms that are most common in the cluster. A simple method is to select the terms with highest weights that appear in all documents. However, this method fails to preserve important terms that are dominant in most (but perhaps not all) documents. The usual preferred alternative is to consider the most frequent indexing units "in the group as a whole" and thus generate "cluster digests" [Cutting et al. 1992]. Using lexical affinities rather than single words as indexing units improves also here the quality of the data, as we are closer to real title than a simple bag of words. Let us consider for instance the top cluster in Figure 6: the top 3 LAs of that cluster according to the method above are "art,danc ; art,salsa ; art, perform" which form what we call a "pseudo-title", as we believe it is easier to infer from this list a real title than from a list of single words (in the example above a list "art, salsa, dance, perform" would not necessarily carry the information about "performing arts".

Yet this method, while in general better than a simple intersection, still has some flaws. This is evidenced in Figure 6, where the LAs "(bean, salsa), (black, salsa), (avocado, salsa)" are so important in the bottom cluster (with cohesion 99%) that they dominate the parent and grandparent, that have exactly the same pseudo-titles. One way to correct this would be to apply this method only on first-level clusters, and at further level, infer a pseudo-title from the direct sub-clusters pseudo-titles only (keeping their respective weights when one cluster has too many sub-clusters). In our example, the middle "bean, salsa' cluster would then have pseudo-title of the form "bean,salsa ; cuisine,salsa" as the latter is the top LA of the singleton cluster "Salsa Mexican Cuisine".

A step further towards real titles would be to extract actual phrases that contain key LAs (rather than simply keywords). One could easily build a utility that would take as input the top 4 LAs shown in Table 2 and extract from the associated page two actual phrases: "Merced County" and "The Yosemite National Park". Similarly, from the pseudo-title "art,danc ; art,salsa ; art, perform", the utility could re-generate two phrases: "art: salsa dancing", and "performing arts", either via extraction techniques or via real natural-language generation ones. We do not consider real summarization (as opposed to snippet extraction) techniques based on the actual contents of each member of the cluster, as this is out of the scope of this paper, but firmly believe that any simple generation tool would work better by taking as input LAs rather than single words.

Finally, another important piece of node-related information is the degree of cohesion under the node (i.e., the similarity value at which the cluster is created). Indeed this value (which ranges between 0 and 1, and is readily available from the clustering algorithm shown in Section 4), can be interpreted easily by users as the minimal degree of similarity among members of the same cluster. Thus, in Figure 6, the cluster labelled 99% gathers two pages that are identical but with very different URLs and were returned by Google, a classical flaw in many search services. One of the reasons for this flaw is that these pages are syntactically different (different banners, base URLs, etc.), while Lassi evaluates the similarity based on profiles. The label "11%" in the cluster from the top in Figure 6, indicates that its members are at least 11% similar. The reader should not confuse the Fetuccino ranking scores shown in Figure 5 that represent the relevance of individual documents to the original query ("salsa"), with the cohesion scores that are assigned only to clusters. Note, however, that the orthogonal ranking information is also represented via the color scheme, where the shade of blue of each document/leaf is derived from the search results ranking score (the darker, the more relevant).

5.2 Presentation of the Clustering Hierarchy

There are numerous visualization methods for representing document hierarchies. We have experimented with various methods and settled on a simple and familiar default tree-control view representation, with the option to switch to various other tree layouts. The rationale for a simple representation is to minimize the learning curve for casual users. The visualization software for Lassi was adapted from the Mapuccino/Fetuccino visualization applet [Maarek et al. 1997], originally designed for representing Web site maps. Thanks to its control/view model, the applet supports various layouts, ranging from a tree-control view as shown in Figure 6 before, to circle view, and even the a"fish-eye view", which enables to emphasize high-level clusters and isolate outliers. The fish-eye view is shown in Figure 7.

Numerous visualization schemes can be used to represent dendograms. Thus, raw tables are used in Scatter/Gather [Cutting et al. 1992] and Grouper[Zamir and Etzioni 1999] for representing just one level of the dendogram. Generic tree or graph layouts as provided by Mapuccino's applet above, or by the more innovative Inxight Hyperbolic Tree^{TM} [Inxight 1999], can easily be customized in order to represent the full dendogram. Fancier visualization paradigms using known metaphors such as landscapes (See ThemeScapes [Wise et. al 95]), or original metaphors such as Cat-a-cone, [Hearst and Karadi 1997], could also map the information contained in the dendogram. The interested reader should consult [Hearst 1999] for a detailed review of graphical overviews of document spaces.

In the context of ephemeral clustering, we believe that a variety of visualization methods can be used, as long as they fulfill the three basic requirements that have been identified before, namely efficiency (if using a Java applet for instance, it should be small enough for the user not to loose patience at downloading time), precision (in a landscape metaphor, proximity should really mean conceptual proximity) and user-friendliness (minimal learning curve and informative interface with which the user can interact).

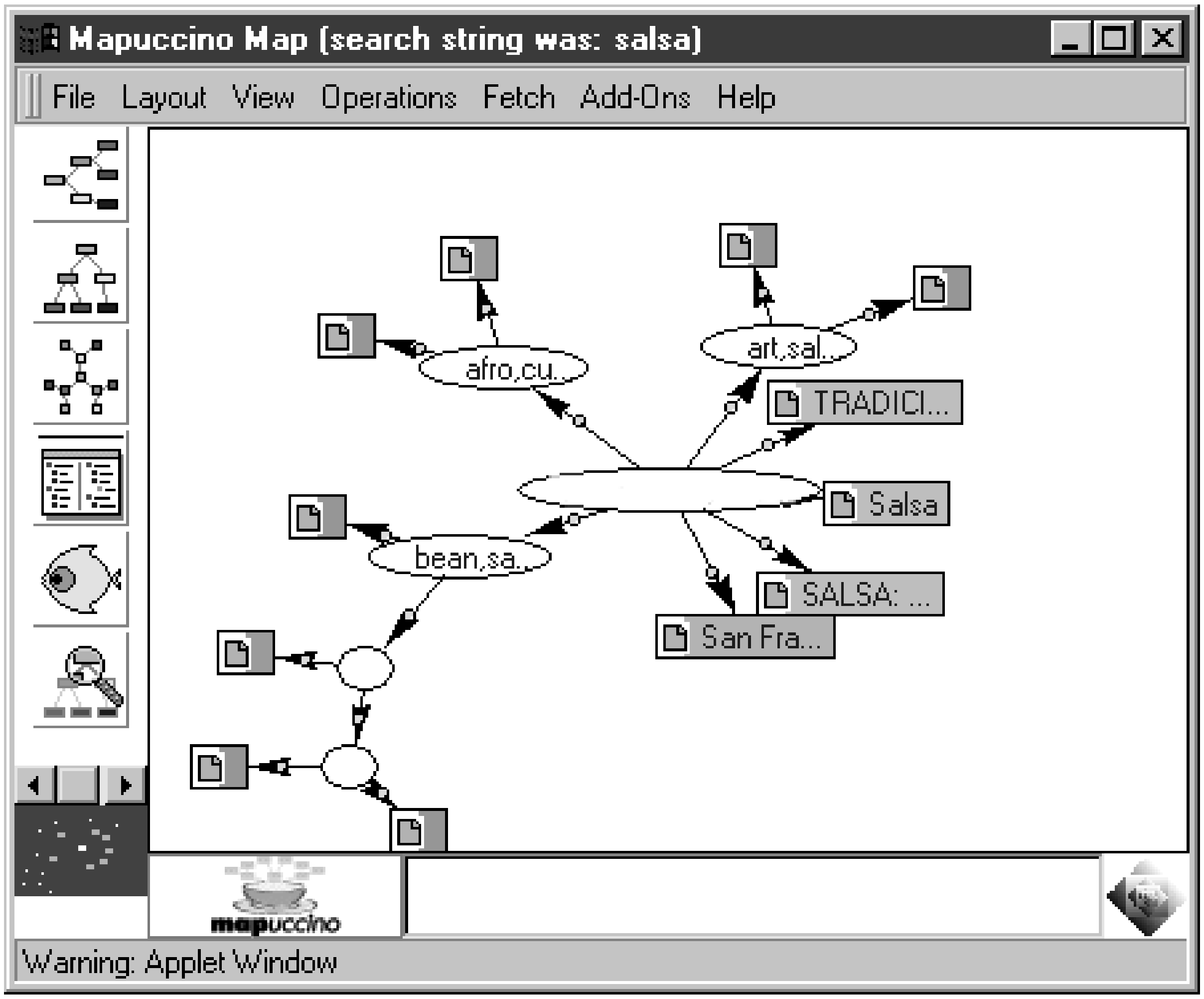


Figure 7: Dendogram in modified cluster view

6 Conclusions

The sheer and ever-growing size of on-line textual content on the Web implies that the value and importance of automatic information retrieval and organization tools is only going to rise in the future. The increase in dynamic content further increases the importance of dynamic, client-based organization. Clustering techniques have been traditionally proposed to assist in the organization of the heavy information "backbone", where content is originated or collected. But the growth in backbone information implies growth in the size of the document sets that are returned as search results, which in turn implies the critical need for client-based tools that organize the results on-line for display and browsing purposes.

Despite similarity in concept, client-based on-line clustering differs substantially in detail from the more common off-line server-based clustering. In particular, it must be lighter, more precise, and provide a presentation layer for interactive use. In this paper we present a set of techniques that collectively address these requirements. Precision is obtained through a novel indexing method that uses pairs of words as the indexing unit. This scheme enhances the quality of the profile, which in turn improves similarity tests and the overall clustering process. Efficiency is obtained through an algorithm that coalesces similar-enough documents or clusters into discrete bins, thereby speeding up the sorting process. This coarse-grained approach also has the desirable side effect of creating hierarchies that are convenient for browsing. Finally, presentation is obtained through a user interface that facilitates cluster-browsing by adding pseudo-titles to internal nodes and through an intuitive visual representation of the hierarchy.

The prospect of client-based on-line organization of documents is far-reaching. One future direction is personalization of clustering based on user and local system preferences, a natural extension of client-based organization. Another direction is an "intermediary" group-level organization, whereby semi-persistent repositories cluster incoming content and hold them for a short period, to serve a group of users with common interests. Our approach enables us to place the clustering functionality anywhere on the path from the origin server to the end user, both spatially and temporally.

Acknowledgments

We wish to thank Ellen Voorhees and David Karger for valuable information on time complexity of HAC algorithms. Thanks to David Bernstein for his "real-life" example of the "merced" query. This research was done while Ron Fagin was a Research Fellow, and Dan Pelleg a visiting graduate student, at IBM Research in Haifa.

References

[Adanson 1757] Adanson, M. Histoire Naturelle du Sénégal. Coquillages. Avec la relation abrégée d'un voyage fait en ce pays, pendant les années 1749,50,51,52 et 53. Bauche, 1757.

[Ben-Shaul et al. 1999] Ben-Shaul, I., Herscovici, M., Jacovi, M., Maarek, Y., Pelleg, D., Shtalhaim, M., Soroka, V., Ur, S. Adding support for dynamic and focused search with Fetuccino. *WWW8 / Computer Networks* 31(11-16), 1999, 1653–1665.

[Bowman et al. 1994] Bowman, C. M., Danzig, P. B., Manber, U., and Schwartz, M. F. Scalable internet: resource discovery. *Communications of the ACM* 37(8), 1994, 98–107.

[Cutting et al. 1992] Cutting, D., Karger, D., Pedersen, J., and Tukey, J., Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Denmark, 1992, 318–329.

[Cutting et al. 1993] Cutting, D., Karger, D., and Pedersen, J., Constant interaction-time scatter/gather/browsing of very large document collections. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, 1993.

[Dom 2000] Dom, B., An Information-Theoretic External Cluster-Validity Measure. IBM Research Report, to appear.

[El-Hamdouchi and Willett 86] El-Hamdouchi, A., and Willett, P. Hierarchical document clustering using Ward's method. In *Proceedings of the 9th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1986, 149–156.

[Everitt 1980] Everitt, B. *Cluster Analysis.* Halsted Press (John Wiley & Sons), New York, 1980.

[Fagan 1989] Fagan, J. The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval. *JASIS* 40(2), 1989, 115–132.

[Frakes and Baeza-Yates, 1992] Frakes, W. and Baeza-Yates, R. (Eds.) *Information Retrieval.* Prentice Hall, NJ, 1992.

[Hearst and Perdersen 1996] Hearst, M. and and Pedersen, J., Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, 1996, 76–84.

[Hearst and Karadi 1997] Hearst, M. and and Karadi, C. Cat-a-cone: An interactive interface for specifying searches and viewing retrieval results using a large category hierarchy. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Philadelphia, PA, 1997, 246–255.

[Hearst 1999] Hearst, M. Chapter 10, User interfaces and visualization. In Baeza-Yates, R. and Ribeiro-Neto, B. eds, *Modern Information Retrieval.* Addison-Wesley, ACM Press, New York, 1999.

[Inxight 1999] http://www.inxight.com/

[Jardine and van Rijsbergen 1971] Jardine, N. and van Rijsbergen, C., The use of hierarchical clustering in information retrieval. *Information Storage and Retrieval* 7(5), 1971, 217–240.

[Lewis 1999] http://www.research.att.com/˜lewis/reuters21578.html

[Maarek 1991] Maarek, Y. Software library construction from an IR perspective. *SIGIR Forum* 25(2), 1991, 8–18.

[Maarek et al. 1991] Maarek, Y., Berry, D., and Kaiser, G. An information retrieval approach for automatically constructing software libraries. *Transactions on Software Engineering* 17(8), 1991, 800–813.

[Maarek and Smadja 1989] Maarek, Y. and Smadja, F. Full text indexing based on lexical relations. an application: software libraries. In *Proceedings of the 12th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Cambridge, MA, 1989, 198–206.

[Maarek and Wecker 1994] Maarek, Y. Wecker, A. The librarian's assistant: automatically assembling books into dynamic bookshelves. In *Proceedings of RIAO'94, Intelligent Multimedia, Information Retrieval Systems and Management*, New York, NY, 1994.

[Maarek and Ben-Shaul 1996] Maarek, Y. and Ben-Shaul, I. Automatically organizing bookmarks per contents. In *Computer Networks and ISDN Systems* Elsevier Science 28, 1996, 1321–1333.

[Maarek et al. 1997] Maarek, Y., Ben-Shaul, I., Jacovi, M., Shtalhaim, M, Ur, S., and Zernik, D. WebCutter: A system for dynamic and tailorable site mapping. *WWW7 / Computer Networks and ISDN Systems* 29(8-13), Elsevier Science, 1997, 1269–1279. Mapuccino applet available for free download at http://www.alphaworks.ibm.com

[Martin et al. 1983] Martin, W., Al, B. and van Sterkenburg, P., On the processing of a text corpus: from textual data to lexicographic information. Lexicography: Principles and Practice R.R.K. Hartmann, (Ed.), London, 1983.

[Michalski and Stepp 1983] Michalski, R., and Stepp, R., Automated constructions of classifications: conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5(4), 1983, 396–409.

[Murtagh 1984] Murtagh, F. *Computational Statistics Quarterly* 1(2), 1984, 101–113.

[Rasmussen 1992] Rasmussen, E. *Information Retrieval, Data Structure and Algorithms*, chapter 16, Clustering Algorithms, Prentice Hall, 1992, 419–442.

[Salton 1971] Salton, G. *The* SMART *retrieval system.* Prentice Hall, Englewood Cliffs, NJ, 1971.

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

[Salton 1989] Salton, G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, Mass, 1989.

[Saussure 1949] Saussure, F. de *Cours de Linguistique Générale.* Librairie Payot, Paris, France, quatrième edition, 1949.

[Shannon] Shannon, C., A mathematical theory of communication. Bell System Technical Journal 27, 379–423, 623–656.

[Voorhees 1986a] Voorhees, E. *The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval.* Ph.D. thesis, Cornell University, 1986.

[Voorhees 1986b] Voorhees, E. *Implementing Agglomerative Hierarchic Clustering Algorithms for Use in Document Retrieval.* TR 86-765, Cornell University, 1986.

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

[Ward 1963] Ward, J. Hierarchical grouping to optimize an objective function. *American Statistical Association* 58(301), 1963, 235–244.

[Wise et. al 95] Wise, J., Thomas, J., Pennock, Lantrip, D., Pottier, M. and Schur, A. Visualizing the non-visual: spatial analysis and interaction with information from text documents. In *Proceedings of the Information Visualization Symposium 95*, IEEE Computer Society Press, 1995, 51–58.

[Zamir and Etzioni 1999] Zamir, O. and Etzioni, O., Grouper: a dynamic clustering interface to web search results. *WWW8 / Computer Networks* 31(11-16), 1999, 1361–1374.