

Grouper: A Dynamic Clustering Interface to Web Search Results

Oren Zamir and Oren Etzioni
Department of Computer Science and Engineering
University of Washington, Box 352350
Seattle, WA 98195-2350 U.S.A.
{zamir, etzioni}@cs.washington.edu
<http://www.cs.washington.edu/research/clustering>

Abstract

Users of Web search engines are often forced to sift through the long ordered list of document “snippets” returned by the engines. The IR community has explored document clustering as an alternative method of organizing retrieval results, but clustering has yet to be deployed on most major search engines. The NorthernLight search engine organizes its output into “custom folders” based on pre-computed document labels, but does not reveal how the folders are generated or how well they correspond to users’ interests.

In this paper, we introduce Grouper – an interface to the results of the HuskySearch meta-search engine, which dynamically groups the search results into clusters labeled by phrases extracted from the snippets. In addition, we report on the first empirical comparison of user Web search behavior on a standard ranked-list presentation versus a clustered presentation. By analyzing HuskySearch logs, we are able to demonstrate substantial differences in the number of documents followed, and in the amount of time and effort expended by users accessing search results through these two interfaces.

Keywords: document clustering, user interface, search engine, visualization

1. Introduction

Conventional document retrieval systems return long lists of ranked documents that users are forced to sift through to find relevant documents. The majority of today's Web search engines (*e.g.*, Excite, AltaVista) follow this paradigm. The notoriously low precision of Web search engines coupled with the ranked list presentation make it hard for users to find the information they are looking for. Our goal is to make search engine results easy to browse by clustering them.

Document clustering has long been investigated as a post-retrieval document visualization technique [1, 4, 13, 18]. Document clustering algorithms attempt to group documents together based on their similarities; thus documents relating to a certain topic will hopefully be placed in a single cluster. This can help users both in locating interesting documents more easily and in getting an overview of the retrieved document set.

Document clustering can be performed, in advance, on the collection as a whole [5], but post-retrieval document clustering has been shown to produce superior results [13]. This is due to the fact that the clusters are computed based on the returned document set; the cluster boundaries are drawn to appropriately partition the set of documents at hand. In contrast, pre-retrieval clusters might be based on features that are infrequent in the retrieved set, as many non-retrieved documents influence the cluster formation.

One way to cluster Web search results is to do so on the search engine. But search engines are under severe resource constraints and dedicating enough CPU time to each query might not be feasible. The clusters, therefore, would have to be pre-computed (at least partially – *e.g.*, NorthernLight). We are interested in exploring an alternative model, one that would not impose additional resource demands on the search engines. We assume the clustering interface is independent of the search engine, *i.e.*, it can reside on a meta-search engine or on a client's browser. Previously we have identified some key requirements for a post-retrieval document clustering interface of this sort [38]:

1. **Coherent Clusters:** The clustering algorithm should group similar documents together. As

documents often have multiple topics, they should not be confined to a single cluster [11]; the clustering algorithm should generate *overlapping clusters* when appropriate.

2. **Efficiently Browseable:** The user needs to determine at a glance whether the contents of a cluster are of interest. Therefore, the system has to provide concise and accurate cluster descriptions.
3. **Speed:** The clustering system should not introduce a substantial delay before displaying the results. This suggests two additional requirements. *Algorithmic Speed:* The clustering algorithm used must be able to cluster hundreds or even thousands of snippets in a few seconds. *Snippet-Tolerance:* As there is no time to download the original documents off the Web, the system should produce high quality clusters even when it only has access to the snippets returned by the search engines.

In earlier work we introduced Suffix Tree Clustering (STC) – a fast, incremental, linear time clustering algorithm that produces coherent clusters [38]. Using traditional IR metrics (*e.g.*, average-precision) we compared STC to other fast clustering algorithms (including k-means [25], Buckshot and Fractionation [5]) on the task of clustering results returned by Web search engines, and showed it to be both faster and more precise than previous algorithms. We used the results to argue that post-retrieval clustering of Web search engine results a-la-STC is feasible.

In this paper, we present Grouper – a clustering interface to the HuskySearch meta-search engine (publicly available at [www.cs.washington.edu/research/clustering]), which uses STC as its clustering algorithm. The paper is organized as follows. First, we describe related work on post retrieval document visualization and document clustering. The following section describes the Grouper system: its user interface and the design decisions made to make the system fast and its clusters easy to browse. Next, we evaluate Grouper in practice based on the logs gathered by this fully deployed system. By contrasting Grouper’s logs with the logs of the standard ranked-list interface to HuskySearch, we are able to demonstrate a substantial difference in the number of documents followed, and in the amount of time and effort expended by users accessing search results through these two interfaces. To the best of our knowledge, this is the first evaluation of a post-retrieval clustering interface on the Web. We conclude by summarizing our contributions and presenting directions for future work.

2. Related Work

The Information Retrieval community has long explored a number of post-retrieval document visualization techniques as alternatives to the ranked list presentation. One category of visualization techniques aims to display additional information about each retrieved document. Often, this will have the secondary effect of grouping documents that share this additional information. The additional information displayed highlights the relationship between the documents and the query terms [10, 33]; predefined document attributes (*e.g.*, size, date, source or popularity) [12, 23]; or user-specified attributes (*i.e.*, predefined topics) [17, 24, 29].

The second category of visualization techniques attempts to visualize inter-document similarities. This can help the user get an overview of the collection, or, alternatively, find interesting documents faster, as once one document has been found it is easy to find others similar to it in the visualized space. There are four major techniques to visualize inter-document similarities: document networks [8, 31], spring embeddings [3, 30], document clustering [1, 13, 18, 38] and self-organizing maps [16, 20, 22]. Of the four major techniques, only document clustering appears to be both fast enough and intuitive enough to require little training or adjustment time from the user [13, 37].

Numerous document clustering algorithms appear in the literature [36]. Agglomerative Hierarchical Clustering algorithms are probably the most commonly used. These algorithms are quadratic in the number of documents and are therefore too slow for our online requirements [34]. Linear time clustering algorithms are the best candidates to comply with the speed requirement of on-line clustering. These include K-Means [25], Single-Pass [14], Buckshot and Fractionation [5]. In earlier work [38] we have introduced another linear time algorithm – STC – which will be briefly described in the next section.

In contrast to STC, most clustering algorithms treat a document as a set of words and not as an ordered sequence of words, thus losing valuable information. Phrases have long been used to supplement

word-based indexing in IR systems [7, 15, 26], yet these methods have not been widely applied to document clustering. The only example known to the authors is the use of the co-appearance of pairs of words as the attributes of the documents' vector representations [21].

Many document clustering algorithms rely on off-line clustering of the entire document collection [6, 28]. Since the Web is a dynamic environment, static, pre-computed clusters would have to be constantly updated, and most clustering algorithms cannot do so incrementally. This would require a huge amount of resources and it has been shown that such an approach results in clusters of lower quality [13]. A pre-computed clustering of a sizeable part the Web (30M HTML documents in over 10 CPU-days) was shown to be feasible [2], but the approach used (a non-hierarchical single-link algorithm with a predetermined similarity threshold) was designed for near-duplicate document identification, and is not suitable for the grouping of retrieval results.

On the Web, there are some attempts to handle the large number of documents returned by search engines. Many search engines provide query refinement features. Excite [www.excite.com], for example, suggests words to be added to or to be excluded from the query. In AltaVista [www.altavista.com], these words are organized into groups, but these groups do not represent clusters of documents.

The NorthernLight search engine [www.northernlight.com] provides "Custom Folders" in which the retrieved documents are organized. These folders can be based on type (*e.g.*, press releases, product reviews, etc.), source (sites, domains), language or subject. The subject-folders are labeled by a short phrase (usually of one or two words). These folders are organized hierarchically and introduce no apparent delay to the search.

NorthernLight does not reveal the method used to create these folders. After experimenting with their system, we hypothesize that at indexing time a set of potential folders is assigned to each document, and at query time the system decides which folders to present. Electric Library [www.elibrary.com] uses "Recurring Themes" in a similar manner. Themes are of three categories: places, persons and subjects. The subject-themes appear to be selected from a restricted, predefined list.

Grouper differs from the NorthernLight approach in several ways. First, it receives hits from different engines, and only looks at the top hits from each search engine. Thus, its clusters are not affected by many low relevance documents, as is the case for NorthernLight's folders. Second, Grouper performs post-retrieval clustering, and thus the clusters are tailored to the retrieved set. Finally, as Grouper can run on a client-machine, it is suitable for distributed IR systems, and addresses the scalability issue.

There are several additional techniques used in Web search engines that are also meant to help the user sift through a large set of retrieved results. Excite, for example, has a "List by Web Site" option that presents an alternative view of the retrieved documents; their "More Like This" option allows, once one interesting document was found, the identification of additional similar documents (though, unlike the other techniques described previously, this performs a new search using the document as a search seed).

3. Grouper

Grouper is a document clustering interface to the HuskySearch meta-search service. HuskySearch (which is based on MetaCrawler [27]) retrieves results from several popular Web search engines, and Grouper clusters the results as they arrive using the STC algorithm. Below we present a brief overview of STC and describe its characteristics. Next, we describe the user interface of the Grouper system. Finally, we explain key design decisions that make Grouper fast, and the clusters easy to understand.

3.1. Overview of the STC Algorithm

STC is a linear time clustering algorithm (linear in the size of the document set) that is based on identifying phrases that are common to groups of documents. A phrase in our context is an ordered sequence of one or more words. We define a base cluster to be the set of documents that share a common phrase.

STC has three logical steps: (1) document "cleaning", (2) identifying base clusters using a suffix tree,

and (3) merging these base clusters into clusters.

In the document "cleaning" step, the string of text representing each document is transformed using a light stemming algorithm. Sentence boundaries are marked and non-word tokens (such as numbers, HTML tags, and most punctuation) are stripped.

The second step – the identification of base clusters – can be viewed as the creation of an inverted index of phrases for our document set. This is done efficiently using a data structure called a suffix tree [9, 35]. This data-structure can be constructed in time linear with the size of the document set, and can be constructed incrementally as the documents are being read [32]. Each base cluster is assigned a score that is a function of the number of documents it contains, and the number of words that make up its phrase. We maintain a stoplist that is supplemented with Internet-specific words (e.g., "previous", "java", "frames" and "mail"). Words appearing in the stoplist, or that appear in too few or too many documents, are not considered as contributing to the score of a base cluster.

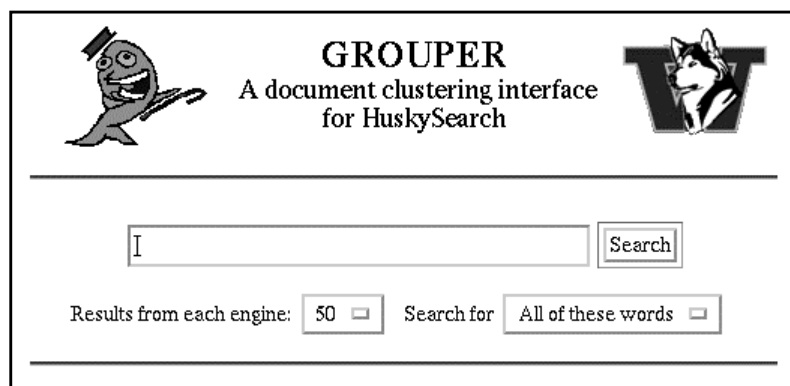
The final step of the STC algorithm merges base clusters with a high degree of overlap in their document sets. This creates clusters that are more coherent semantically by allowing documents to be in the same cluster even if they do not share a common phrase but rather share phrases with other documents of the cluster. This step also reduces the fragmentation of the produced clusters.

The STC algorithm has several novel characteristics. It creates *overlapping clusters*, i.e., a document can appear in more than one cluster. This is in contrast to most other clustering algorithms, which force each document into a single cluster. STC uses phrases to identify similarities between documents and consequently to construct its clusters. Most other clustering algorithms treat a document as a bag of words and disregard the additional information that is present in the ordering of the words. The shared phrases of a cluster also provide an informative way of summarizing its contents to the user.

The STC algorithm is particularly well suited for Web document clustering, as it is both fast and incremental. Queries on the Web often yield a large number of loosely related documents, and often many snippets are returned that do not show any correlation to the query. STC has been shown robust in such "noisy" situations. It also does not coerce the documents into a predefined number of clusters, thus allowing the documents themselves to determine the number of clusters generated.

3.2. User Interface

A Grouper session starts with the user entering her query in the query box (Figure 1). The user can choose how query terms are treated (e.g., as a phrase, etc.), and can specify the number of documents to be retrieved (10 – 200) from each of the participating search engines. As the system queries approximately 10 search engines, it will typically retrieve 70 – 1000 documents, after eliminating duplicates. After all search engines have returned (or 10 seconds have passed), the main results page is displayed (Figure 2).



The screenshot shows the Grouper web interface. At the top left is a cartoon character holding a wrench. In the center, the text reads "GROUPER" followed by "A document clustering interface for HuskySearch". To the right is a husky logo. Below this is a search input field containing the letter "I" and a "Search" button. At the bottom, there are two dropdown menus: "Results from each engine:" with "50" selected, and "Search for:" with "All of these words" selected.

Figure 1: Grouper's query interface. Users do not need to enter any parameters for the clustering algorithm.

The main results page displays the number of documents retrieved and the number of clusters found. The clusters are presented in a large table – each cluster in a single row referred to as the *summary* of the cluster. The clusters are ordered by their estimated coherence. A summary of a cluster includes its size (the number of documents it contains), and attempts to convey the content of the documents in the clusters by displaying *shared phrases* and *sample titles*. Shared phrases are phrases that appear in many documents of the cluster. The numbers appearing in parenthesis after each phrase indicate the percentage of the documents in the cluster that contain the phrase. Titles of three sample documents are also displayed in the summary of each cluster.

If a summary of a cluster indicates to the user that the cluster is of interest, the user can take several actions. If a certain title seems promising, the user can go directly to that document by clicking on the title. Alternatively, the user can click on the "View Results" links to go to the cluster's page. There, the snippets of all the documents in the cluster are presented in a ranked list.

Query: israel		
Documents: 272, Clusters: 15, Average Cluster Size: 15.1 documents		
Cluster	Size	Shared Phrases and Sample Document Titles
1 View Results Refine Query Based On This Cluster	16	Society and Culture (56%), Faiths and Practices (56%), Judaism (69%), Spirituality (56%); Religion (56%), organizations (43%) ● Ahavat Israel - The Amazing Jewish Website! ● Israel and Judaism ● Judaica Collection
2 View Results Refine Query Based On This Cluster	15	Ministry of Foreign Affairs (33%), Ministry (87%) ● Publications and Data of the BANK OF ISRAEL ● Consulate General of Israel to the Mid-Atlantic Region ● The Friends of Israel Gospel Ministry
3 View Results Refine Query Based On This Cluster	11	Israel Tourism (36%), Comprehensive Israel (36%), Tourism (64%) ● Interactive Israel tourism guide - Jerusalem ● Ambassade d'Israel ● Travel to Israel Opportunites
4 View Results Refine Query Based On This Cluster	7	Middle East (57%), History (57%); WAR (42%), Region (42%), Complete (42%), Listing (42%), country (42%) ● Israel at Fifty: Our Introduction to The Six Day War ● Machal - Volunteers in the Israel's War of Independence ● HISTORY: The State of Israel
5 View Results Refine Query Based On This Cluster	22	Economy (68%), Companies (55%), Travel (55%) ● Israel Hotel Association ● Israel Association of Electronics Industries ● Focus Capital Group - Israel

Figure 2: The main results page in Grouper for the query “israel”. Each row in the table is the *summary* of a cluster – an attempt to convey the content of the documents in the cluster. It includes the size of the cluster, *shared phrases* – phrases that appear in many documents of the cluster, and up to three *sample titles* of documents in the cluster. The numbers appearing in parenthesis after each phrase indicate the percentage of the documents in the cluster that contain the phrase. In the example above only the first five clusters are shown.

Finally, the user may want to retrieve additional similar documents. Several search engines (*e.g.*, Excite) allow the user to use a certain document as a seed for a new search (the "More Like This" option). However, this method does not allow the user to control the search being made. We wanted to give the user the ability to find documents that are related to the cluster of interest, but allow the user to control the exact search that is being done. The "Refine Query Based On This Cluster" option in Grouper is designed to do precisely that (Figure 3) – the user can use the phrases that characterize the cluster to reformulate the query. To our knowledge, Grouper is the only search engine that suggests multi-word phrases to the user for query refinement. Looking at the logs of the system, we see that in 80% of the cases where users used the “Refine Query” option, they used the phrases suggested by the system, and 41% of these were multi-word phrases. These figures seem to suggest that users find this feature beneficial in practice.

Want to be more specific?
 Use the phrases found to focus your search!
 Click on the phrases and/or words you would like to add to your search.
 Then click on the search button.

Results from each engine:
Search for

"Society and Culture"
 "Faiths and Practices"
 Judaism
 Spirituality
 Religion
 organizations

Figure 3: The query refinement page for the first cluster of the query “israel”. The user can reformulate the query using the phrases that were identified in the cluster. Clicking on a checkbox associated with a phrase will include it in the refined query.

The Grouper search service has been running in its current format since July 1998, and receives approximately 65 queries a day. We are logging all activities on this search engine and have logged to date more than 10,000 queries.

A noteworthy characteristic of STC is that the user does not need to input the required number of clusters to the system. The clusters produced are ordered by their approximate coherence, and the clusters gracefully degrade from coherent clusters at the top of the cluster list, to clusters that are characterized by a single phrase or word at the bottom of the list. These clusters do not necessarily represent a semantically coherent topic, but rather function as an index to the retrieved document set. We believe these *index-clusters* are useful to the user when no earlier cluster appears to be of interest. This sometimes occurs when there are no “natural” clusters in the retrieved document set or when the user’s interest is narrow and is overshadowed by more dominant themes among the retrieved documents.

Analyzing the system’s logs has showed that in approximately 21% of the sessions users followed documents from index-clusters. We find this figure to confirm our reasoning in including this type of clusters in our output.

3.3. Making the Clusters Easy to Browse

As mentioned in the introduction, it is not enough for a clustering system to create coherent clusters, but the system must also convey the contents of the clusters to the users concisely and accurately. The system is most useful when the user can decide at a glance whether the contents of a cluster are of interest.

One approach of describing a cluster includes presenting words that appear frequently in the documents of the cluster (the cluster’s centroid) as well as presenting the titles of selected documents [13]. As STC creates clusters based on phrases that are shared by many of their documents, Grouper can also use these phrases to describe the cluster. We have found these phrases to be extremely useful in characterizing the cluster’s content.

Given a set of phrases that define an STC cluster we present them in descending order of coverage (the percent of documents in the cluster that contain the phrase) and length (number of words in the phrase, not counting stopped words nor words appearing in the query). Since each cluster can contain many phrases, we display at most the first six phrases¹ because our goal is to create a compact cluster summary. We have found that in many cases some of the displayed phrases are quite similar and therefore redundant. This reduces the conciseness and clarity of the summary, as well as prevents additional informative phrases from being displayed. Such redundant phrases are therefore not displayed. There are three heuristics we use to identify redundant phrases:

1. **Word Overlap:** If a phrase has more than 60% of its non-stopped words appearing in another phrase

¹ The fifth and the sixth phrases are displayed only if their coverage is greater than 50%.

of higher coverage, it will not be displayed. In Table 1, phrase 7 is not displayed as 75% of its words appear also in phrase 6, which also had a higher coverage.

2. **Sub- and Super- Strings:** Often, we find that STC identifies phrases that are sub-strings of other phrases (we call these *sub-phrases* and *super-phrases*). This happens both when the two phrases are independent phrases in their own right (such as “united states” and “president of the united states”) and when the phrase was truncated in the snippet received from the search engine (such as “president of the united states” and “president of the united...”). A sub-phrase will always have a higher coverage than its super-phrase, but the super-phrase will be more specific and therefore more informative. To balance this tradeoff, we determine, for each phrase, whether it has a sub-phrase or a super-phrase among the other phrases of the cluster. If it doesn’t have a sub-phrase, it is designated as a *most-general* phrase; if no super-phrase exists, it is designated as a *most-specific* phrase. We will consider phrases redundant if they are not in either of these categories. In Table 1, phrases 3 and 4 will not be displayed, as they are neither most-general nor most-specific.
3. **Most-General Phrase with Low Coverage:** The purpose of displaying short, general phrases is to achieve high coverage. In cases where a most-general phrase adds little to the coverage of its corresponding most-specific phrase, we will not display it. Thus, we define a minimal coverage difference (we found 20% to be satisfactory) that must be attained in order for the most-general phrase to be displayed. In Table 1, phrase 8 will not be displayed, even though it is a most-general phrase, as it adds only 5% to the coverage of phrase 6.

Num.	Phrase	Coverage	Most-Spec.	Most-Gen.	Selected
1	earth summit	60%	+	+	✓
2	vice president of the united states of america	30%	+		✓
3	president of the united states of	40%			
4	united states of america	50%			
5	united states	65%		+	✓
6	greenhouse gas emissions forecast	40%	+		✓
7	reducing emissions of greenhouse gas	30%	+		
8	greenhouse gas	45%		+	

Table 1: Determining which phrases to display: The table shows the set of phrases identified in a cluster. The phrases are presented along with their coverage, whether they are most-specific or most-general and whether they are selected to be displayed. Phrases 3 and 4 are redundant, and thus will not be displayed, as they are neither most-general nor most-specific (the *Sub- and Super- Strings* heuristic). Phrase 7 is redundant as 75% of its words also appear in the phrase 6, which also has a higher coverage (the *Word Overlap* heuristic). Phrase 8 is redundant, even though it is a most-general phrase, as it adds only 5% to the coverage of phrase 6 (the *Most-General Phrase with Low Coverage* heuristic).

As mentioned earlier, documents are processed before they are inserted into the suffix tree (step 1 of STC). This processing improves the identification of common phrases, however the readability of the text is reduced. To deal with this problem we keep the original (unprocessed) text of the documents; instead of displaying a processed, and less comprehensible, phrase, we display the original text that corresponds to it. For example the phrase "post retrieve document cluster" might actually correspond to the string "post-retrieval document clustering", which might be more meaningful to the user.

In addition to describing the cluster using its phrases, we also use the standard technique of identifying and presenting single words that appear frequently within the documents of the cluster. We display up to five such words for each cluster. Words will not be shown if they appear in less than 40% of the documents, or in the stoplist, or any of the displayed phrases. A semi-colon separates these words from the phrases of the cluster. There is one major difference between the phrases and these frequent words that should be emphasized. A phrase indicates that all the documents containing it are in the

cluster, because it was a basic blocks on which the cluster was constructed. Documents containing a frequent word, on the other hand, might not all appear in the cluster.

Another design decision in Grouper was how to order the documents in a cluster. This affects both the order of the documents on the cluster's page, and the choice of the three documents that have their titles displayed in the cluster's summary on the main results page. Two options were considered: sorting the documents based on their relevance to the query or sorting them based on the number of the cluster's phrases each contains (this is similar to the option of sorting the documents of a cluster based on their similarity to the query or based on their similarity to the cluster's centroid, as described in [13]). We chose the second approach – sorting by the number of the cluster's phrases each document contains – as we believe this results in more informative cluster summaries.

3.4. Design for Speed

Speed is critical for any Web search interface. Grouper has three characteristics that make it fast: incrementally of the clustering algorithm, efficient implementation, and the ability to form coherent clusters based on the snippets returned by search engines. We discuss each characteristic in turn below.

Because STC is incremental, Grouper can use the "free" CPU time while it is waiting for snippets to arrive over the Internet from the search engines invoked by HuskySearch. As a result, Grouper typically produces results immediately after the last document arrives, whereas a non-incremental algorithm would only start its computations at that point.

There are two possible modes of clustering Web search results. The system can either respond in seconds by clustering the snippets returned by the search engine, or it can download the original documents off the Web and cluster them, requiring more time as downloading the documents can be quite slow. On the other hand, the clustering quality of the latter mode is higher since more information is present. In previous work, we investigated this tradeoff and found this degradation in the quality of the clusters to be moderate (~15% for STC using the average-precision metric) [38]. Grouper therefore performs the clustering on the returned snippets, allowing fast interaction with the user².

The STC algorithm used in Grouper is linear in the number of documents. In addition, we used several implementation techniques to make it efficient in practice. STC performs a large number of string comparisons; to do this efficiently, each word is transformed into a unique integer and thus faster integer comparisons could be used. The suffix tree itself was designed as a tree of words, not of characters. As there is a large variability in the branching factor of the nodes of the suffix tree, different tree implementations were used for nodes of different branching factors. To allow efficient calculation of document overlap between base clusters, the documents of each base cluster were encoded as a bit-vector.

Additional speedup was achieved by realizing that we were not interested in all possible phrases. Phrases that begin or end with a word appearing in our stoplist have the same semantic meaning without these stopped words, and therefore these stopped words should be stripped (for example the phrase “the vice president of” should be stripped to “vice president”, while the phrase “the vice president of the US” should be stripped to “vice president of the US”). Therefore, when inserting a string into the suffix tree, all leading and ending stopped words can be disregarded. Moreover, as we are only interested in phrases that appear in more than a certain number of documents, we can also strip off words that do not appear in this minimal number of documents. These improvements reduced the workload by more than 50%.

These improvements help the system reach the speed required for an interactive, online search engine. In Figure 4 we present the time it takes Grouper to cluster the retrieved documents as a function of the number of documents retrieved. We recorded the clustering time of 4076 Grouper queries, and grouped them by the number of documents retrieved (by steps of 50); the time reported is the time required by the clustering module. As our machine (a DEC AlphaStation 500, 333Mhz with 512M RAM)

² For users willing to wait, Grouper also has a “Quality Search” option that will download the original documents off the Web and use them in the STC algorithm. This option is much slower (1-5 minutes as opposed to a few seconds), but will generally produce better results.

is shared among several research projects, the reported times are greatly affected by its load as well as the load on our file server, and would be improved substantially by allocating dedicated hardware to Grouper.

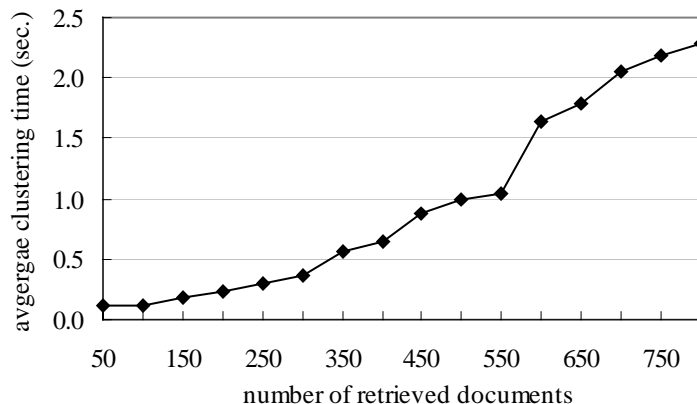


Figure 4: The average time (in seconds) it takes Grouper to cluster the retrieved documents as a function of the number documents retrieved.

4. Empirical Evaluation of Grouper

The evaluation of a clustering interface is notoriously difficult, particularly in the context of Web search engines, which are used by a heterogeneous user population for a wide variety of tasks: from finding a specific Web document that the user has seen before and can easily describe, to obtaining an overview of an unfamiliar topic, to exhaustively examining a large set of documents on a topic, and more. A clustering system will prove useful only in a subset of these cases.

We had previously shown that the STC algorithm produces coherent clusters, and actually outperforms, in this respect, other clustering algorithms in the Web search domain [38]. This was demonstrated by comparing the different algorithms to each other and to the ranked-list presentation using the standard IR measure of average-precision. To apply this metric, we developed a model of the user's use of the clustering results and had to create our own relevance judgments for search results.

In this paper, instead of evaluating Grouper under similar assumptions, or in a laboratory user study, we chose to utilize the logs of the deployed system "in action" as the basis of our evaluation. The logs record the behavior of a large number of Grouper and HuskySearch users on queries of their choice. . The Grouper logs were recorded between 15/9/98-15/11/98, and represent 3183 queries. The HuskySearch logs were recorded between 26/10/98-09/11/98, and represent 19330 queries.

4.1. Coherent Clusters

The first hypothesis we investigated through log analysis is that users will tend to follow documents from relatively few clusters. First, we calculated the average number of followed clusters (clusters from which a user had followed documents) as a function of the number of documents followed thus far in the session. Next, we compared this to the number of followed clusters if a random clustering had been created (using the same number of clusters and cluster size distribution).

It can be argued that by presenting the documents organized in clusters, we bias the user to follow multiple documents from the same cluster. We therefore performed the following experiment as well. We recorded which documents were followed by HuskySearch users. Next, We clustered these documents and, assuming the same documents would have been followed, we calculated the average number of clusters that would have been followed in these HuskySearch sessions. To provide a benchmark for STC's performance on the HuskySearch data, we compared STC with the popular K-Means clustering

algorithm (Figure 5)³.

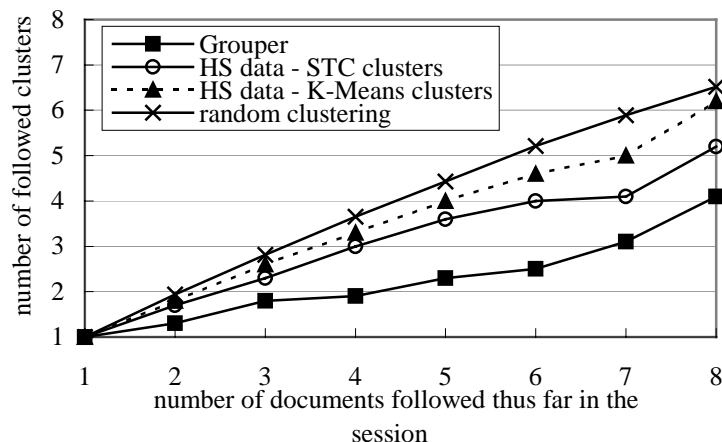


Figure 5: The average number of followed clusters as a function of the number of documents followed thus far in the session. We compare Grouper clusters to the number of followed clusters if a random clustering had been created, and show that Grouper is substantially better than random. In addition, we present the calculated number of clusters followed using HuskySearch data: the documents retrieved in HuskySearch search sessions are clustered using both STC and K-Means; assuming that the same documents would have been followed, we calculate the number of followed clusters. We see that STC produces slightly more coherent clusters than K-Means.

We had hypothesized that the user would visit few clusters in Grouper, but the numbers are not as low as we had hoped (*e.g.*, users viewing seven documents visit three clusters on average). This result can be interpreted in several ways. It might indicate that even when users follow many documents, they typically seek an overview of the retrieved results and not necessarily an exhaustive review of all the documents on a specific subtopic. Alternatively, these figures might indicate that our clustering algorithm (using the documents' snippets) falls short of the ideal goal of creating a single cluster corresponding to the user's information needs.

4.2. Comparison to a Ranked List Display

In this section, our goal is to compare the Grouper clustering interface to the traditional ranked-list interface available for HuskySearch. HuskySearch and Grouper are identical (including the machines they run on) with the exception of their result presentation

Having recorded user behavior on both Grouper and HuskySearch, we would ideally report whether users were able to find more relevant information, and do so more efficiently, using Grouper's clustering interface as compared with the ranked-list presentation in HuskySearch. However, since -in contrast with TREC experiments - we do not know the exact information need or the set of relevant documents, we cannot rely on the standard notions of precision and recall. Instead, we decided to focus on the information collected in our user logs, and used the following metrics in our evaluation, each of which is subject to multiple caveats which we mention below:

1. **Number of documents followed:** we recorded the number of documents "clicked on" by users as an indication of the "amount" of information users were retrieving (Table 2). However, clicks are a coarse measure of the information obtained; one click may indicate mild interest whereas another may indicate that the user "struck gold". Worse, people may be seduced by a snippet into clicking on a document that is of no interest. Finally, users may obtain valuable information directly from

³ We have compared STC with additional clustering algorithms (Buckshot, Fractionation, and Agglomerative Hierarchical Clustering) and found similar results.

- snippets or cluster summaries without any clicks.
2. **Time spent:** as a rough measure of search efficiency, we recorded the amount of time users spent traversing the results set (Figure 6a). The main problem with this measure is that the time recorded is the *sum* of time spent in network delays, in reading documents, and in traversing the results presentation (ranked list or clusters). Our server-based experimental apparatus cannot decompose the time measured into its components.
 3. **Click distance:** as another rough measure of search efficiency, we recorded the distance between successive user clicks on the document set (Figure 6b). While in the ranked-list presentation, this notion has an unambiguous interpretation (the distance between the first document in the list and the tenth document is nine), it's not immediately obvious what is the corresponding notion in the cluster presentation. Specifically, when a user skips a cluster, what is the distance traversed? We discuss our distance function below.

We acknowledge that in the experiments described below, we are comparing the behavior of two distinct user populations – HuskySearch users and Grouper users. Nevertheless, the data we collected shows substantial differences in how people use the two interfaces. We plan to follow up with a carefully controlled user study that will complement the log analysis and help to test hypotheses as to *why* the observed differences arise.

Looking at the number of retrieved documents that were followed by the user, we note a considerable difference between the two systems. In HuskySearch users followed only 1.0 documents on average, whereas in Grouper, users followed 40% more. Table 2 presents the percentage of sessions in which the user has followed a certain number of retrieved documents for both systems. All sessions resulting in 8 or more followed documents are presented in a single column (the “8+” column). The data shows that the percentage of sessions in which users do not follow even a single document is smaller in Grouper (46%) than in HuskySearch (53%), while the percentage of sessions in which users followed multiple documents is higher in Grouper.

Optimistically, Table 2 suggests three possible hypotheses: First, it might be easier to find interesting documents using Grouper (as fewer sessions result in no document being followed). Second, once an interesting document has been located, Grouper seems to do a better job in helping the user find additional interesting documents. And third, users might prefer a clustering interface such as Grouper when faced with tasks where several documents are required. As mentioned earlier, a user study is needed to test these hypotheses.

Num. of Docs. Followed:	0	1	2	3	4	5	6	7	8+
% of HuskySearch sessions	53.0	26.9	8.4	4.2	2.3	1.6	1.1	0.7	1.9
% of Grouper sessions	46.0	25.2	10.2	6.0	3.9	2.4	1.8	1.4	3.2

Table 2: The percentage of sessions in which users have followed a certain number of retrieved documents in Grouper and in HuskySearch. All session resulting in 8 or more followed documents are presented in a single column (the “8+” column). In HuskySearch, users followed two or more documents in 20% of the sessions, while in Grouper they did so in 29% of the sessions.

The second metric we use to compare the two systems is the time the user spends on each document followed. When a user follows a document, the request is actually through our server and we log the arrival times of these requests. The time spent on a followed document is measured as the time between a user’s request for a document and the user’s previous request (or, for the first followed document, the time the results page was sent to the user). The measured time therefore includes the time to download and view a selected document and the time to find the next document of interest. We exclude from the analysis all sessions in which more than 200 seconds elapsed between the following of consecutive document (approximately 10% of the sessions) as the user might have abandoned the search task for a while. The time spent on each document as a function of the number of the documents followed thus far in the session is presented in Figure 6a.

From Figure 6a it can be seen that the average time per document for the first three followed documents of a session is lower in HuskySearch, but the Grouper interface is apparently faster for the rest of the followed documents of the session.

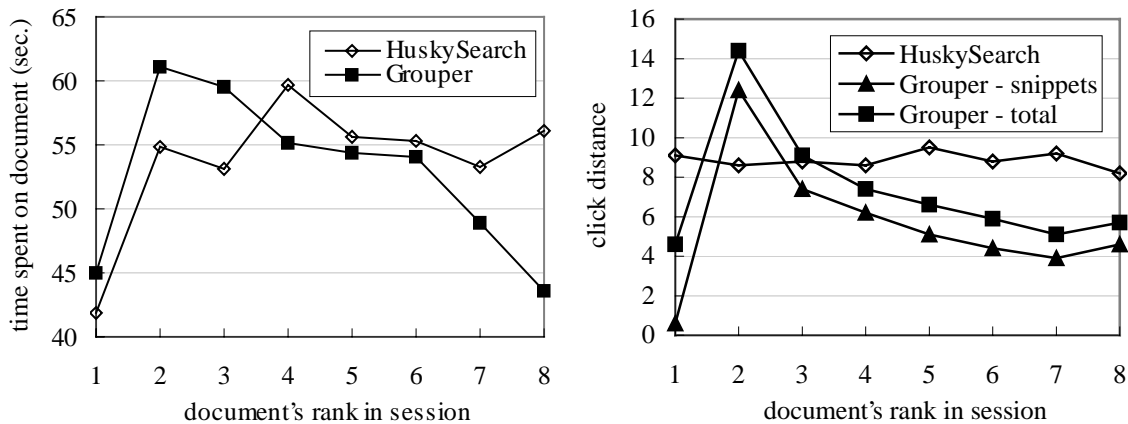


Figure 6a: The average amount of time spent on each document followed, as a function of the document's rank (the number of documents followed thus far in the session). **Figure 6b:** The click distance as a function of the document's rank. This metric attempts to capture the effort spent by the user to find the next interesting document. The "Grouper-snippets" line represents the number of snippets skipped. The "Grouper-total" line represents the total click distance, which is the sum of the number of snippets and clusters skipped (under the assumption that the "cost" of scanning a cluster's summary is equal to that of scanning a snippet).

While speed appears to be an advantage for Grouper when multiple documents are followed, some caveats are in order. In the analysis above we assume the time to download a document and the time to view it are independent of the system used and of a documents' rank in the session, and therefore these factors should average out across all measurements. Unfortunately, we cannot validate this empirically in our system. Another possible problem in interpreting this metric is that it is not clear that shorter time is better. It might indicate that an interesting document was easier to find, but on the other hand, it might indicate that the user did not find the document interesting enough to dedicate much time to it. Again, the logs do not give us enough information to discriminate between these possible interpretations.

The third metric used to compare the two systems was click distance, an attempt to estimate the cost to the user of finding the next interesting document. We assume only documents followed are beneficial to the user; snippets that are scanned but not followed represent wasted efforts. Likewise, the user is assumed to gain nothing from the structure of the clusters. The distance between clicks on a ranked list is simply the number of snippets between the two clicks (or proceeding the first click). In the cluster interface, we assume the user does not scan through the snippets in clusters that are skipped, but in any cluster visited, all of the snippets are scanned. For example, if the user clicks on the second document of the first cluster (which contains 20 documents), skips the second cluster and then clicks on the fifth document of the third cluster, we assume she scanned 22 snippets between these two clicks (18 in the first cluster and 4 in the third). In the clustering interface there is the additional cost of skipping clusters. This should be added to the measure presented above as they both represent wasted effort.

This data is presented in Figure 6b. The "Grouper-snippets" line represents the number of snippets skipped. The "Grouper-total" line represents the total click distance, which is the sum of the number of snippets and clusters skipped. We assume the "cost" of scanning a summary of a cluster is equal to that of scanning a snippet. Of course, alternative assumptions can be made, and it is easy to see from the graph how these alternative "Grouper-total" lines would look. These results exhibit a trend similar to the results of the previous experiment – finding the first few interesting documents actually requires more "effort" in Grouper as compared to HuskySearch's ranked-list presentation, but after the first two or three

documents, finding additional interesting documents appears to require less effort in Grouper. This possibly reflects the fact that the user must spend some time/effort to understand the clusters that Grouper presents, but after doing so the clusters are helpful in finding required information faster. It also confirms our observation that a clustering interface is not suitable for all search tasks.

5. Conclusion

In this paper, we have presented Grouper – a clustering interface for Web search engine results. We described the system and the design decisions made in its implementation. Grouper enabled us to carry out the first empirical assessment of user behavior given a clustering interface to Web search results. By analyzing the system's logs and comparing them to the logs of HuskySearch, we were able to show substantial difference in the patterns of use of the two systems. We are now planning a controlled user study to validate some of the hypotheses inferred from this data.

By fielding a document clustering system on the Web, we were also able to identify some of its shortcomings. Grouper creates clusters by merging base clusters (a phrase and the set of documents that contain it). This is often beneficial, but it can be confusing, especially when the clusters fail to capture the semantic distinctions the users were expecting. Grouper-II will address this issue by allowing users to view non-merged base clusters. This view is essentially an inverted index of phrases to the retrieved document set. We believe this view will be useful in a higher portion of the queries and will be more intuitive to novice users.

We have also noticed that the number of clusters created increases as more documents are retrieved. As we are interested in making the system scale to substantially larger retrieved document sets, clusters should be presented hierarchically so users can navigate the results more efficiently. Grouper-II will support a hierarchical and interactive interface, similar to Scatter/Gather [13]. The user will be able to select one or more base clusters to focus on, thus defining a subset of the retrieved document set. Grouper-II will then, recursively, display the base clusters found in this subset of documents.

Acknowledgements We thank AnHai Doan, Marc Friedman, Mike Perkowitz and Erik Selberg for commenting on earlier draft of this paper. This research was funded in part by Office of Naval Research grant 98-1-0177, and by National Science Foundation grants IRI-9357772 and DL-9874759.

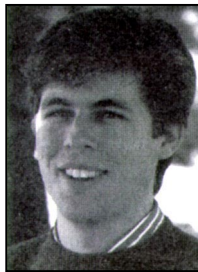
6. References

- [1] R. B. Allen, P. Obry and M. Littman, An interface for navigating clustered document sets returned by queries, in: *Proceedings of the ACM Conference on Organizational Computing Systems*, 1993, pp 166-171.
- [2] A. Z. Broder, S. C. Glassman, M. S. Manasse and G. Zweig, Syntactic clustering of the Web, in: *Proceedings of the Sixth International Web Wide World Conference (WWW6)*, 1997.
- [3] M. Chalmers, and P. Chitson, BEAD: explorations in information visualization, in: *Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92)*, 1992, pp 330-337.
- [4] W. B. Croft, *Organizing and searching large files of documents*, Ph.D. Thesis, University of Cambridge, 1978.
- [5] D. R. Cutting, D. R. Karger, J. O. Pedersen and J. W. Tukey, Scatter/Gather: a cluster-based approach to browsing large document collections, in: *Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92)*, 1992, pp 318-329.
- [6] D. R. Cutting, D. R. Karger and J. O. Pedersen, Constant interaction-time Scatter/Gather browsing of large document collections, in: *Proceedings of the 16th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*, 1993, pp 126-135.
- [7] J. L. Fagan, Experiments in automatic phrase indexing for document retrieval: a comparison of syntactic and non-syntactic methods, Ph.D. Thesis, Cornell University, 1987.

- [8] R. H. Fowler, W. A. Fowler and B. A. Wilson, Integrating query, thesaurus, and documents through a common visual representation, in: *Proceedings of the 14th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'91)*, 1991, pp 142-151.
- [9] D. Gusfield, Algorithms on strings, trees and sequences: Computer Science and Computational Biology, chapter 6, Cambridge University Press, 1997.
- [10] M. Hearst, TileBars: visualization of term distribution information in full text information access, in: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'95)*, 1995, pp 59-66.
- [11] M. A. Hearst, The use of categories and clusters in information access interfaces, in: T. Strzalkowski (Ed.), *Natural Language Information Retrieval*, Kluwer Academic Publishers, 1998.
- [12] M. Hearst and C. Karadi, Cat-a-Cone: an interface for specifying searches and viewing retrieval results using a large category hierarchy, in: *Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, 1997.
- [13] M. A. Hearst and J. O. Pedersen, 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 (SIGIR'96)*, 1996, pp 76-84.
- [14] D. R. Hill, A vector clustering technique, in: Samuelson (Ed.), *Mechanized Information Storage, Retrieval and Dissemination*, North-Holland, Amsterdam, 1968.
- [15] D. A. Hull, G. Grefenstette, B. M. Schulze, E. Gaussier, H. Schütze and L. O. Pedersen, Xerox TREC-5 site report: routing, filtering, NLP, and Spanish tracks, in: D. K. Harman (Ed.), *The Fifth Text Retrieval Conference (TREC-5)*, NIST Special Publication, 1997.
- [16] T. Kohonen, Exploration of very large databases by self-organizing maps, in: *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1, 1997, pp 1-6.
- [17] R. Korfhage, To see, or not to see – is that the query?, in: *Proceedings of the 14th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'91)*, 1991, pp 134-141.
- [18] A. V. Leouski and W. B. Croft, An evaluation of techniques for clustering search results. Technical Report IR-76, Department of Computer Science, University of Massachusetts, Amherst, 1996.
- [20] Lin, X., A self-organizing semantic map for information retrieval, in: *Proceedings of the 14th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'91)*, 1991, pp 262-269.
- [21] Y. S. Maarek and A. J. Wecker, The Librarian's Assistant: automatically organizing on-line books into dynamic bookshelves, in: *Proceedings of the International Conference on Intelligent Multimedia Information Retrieval Systems and Management (RIA0'94)*, 1994.
- [22] D. Merkl, Exploration of text collections with hierarchical feature maps, in: *Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, 1997, pp 186-195.
- [23] L. T. Nowell, R. K. France, D. Hix, L. S. Heath and E. Fox, Visualizing search results: some alternatives to query document similarity, in: *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, 1996, pp 67-75.
- [24] K. A. Olsen, R. R. Korfhage, M. B. Spring, K. M. Sochats and J. G. Williams, Visualization of a document collection: the VIBE system. *Information Processing and Management*, vol. 29(1), 1993, pp 69-81.
- [25] J. J. Rocchio, Document retrieval systems – optimization and evaluation, Ph.D. Thesis, Harvard University, 1966.
- [26] G. Salton, C. S. Yang and C. T. Yu, A theory of term importance in automatic text analysis, *Journal of the American Society for Information Science*, vol. 26(1), 1975, pp 33-44.
- [27] E. Selberg and O. Etzioni, Multi-service search and comparison using the MetaCrawler, in: *Proceedings of the 4th World Wide Web Conference (WWW4)*, 1995.
- [28] C. Silverstein and J. O. Pedersen, Almost-constant time clustering of arbitrary corpus subsets, in: *Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, 1997, pp 60-66.
- [29] A. Spoerri, InfoCrystal: A visual tool for information retrieval and management, in: *Proceedings of Information Knowledge and Management (CIKM'93)*, 1993, pp 150-157.
- [30] R. Swan and J. Allan, Aspect Windows, 3-D Visualizations, and Indirect Comparisons of Information Retrieval System, in: *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, 1998.

- [31] R. H. Thompson and W. B. Croft, Support for browsing in an intelligent text retrieval system. *International Journal of Man-Machine Studies*, vol. 30(6), 1989, pp 639-668.
- [32] E. Ukkonen, On-line construction of suffix trees, *Algorithmica*, vol. 14, 1995, pp 249-260.
- [33] A. Veerasamy and N. J. Belkin, Evaluation of a tool for visualization of information retrieval results, in: *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, 1996, pp 85-92.
- [34] E. M. Voorhees, Implementing agglomerative hierarchical clustering algorithms for use in document retrieval, *Information Processing and Management*, vol. 22, 1986, pp 465-476.
- [35] P. Weiner, Linear pattern matching algorithms, in: *Proceedings of the 14th Annual Symposium on Foundations of Computer Science (FOCS)*, 1973, pp 1-11.
- [36] P. Willet, Recent trends in hierarchical document clustering: a critical review. *Information Processing and Management*, vol. 24, 1988, pp 577-597.
- [37] O. Zamir, Visualization of search results in document retrieval systems, General Examination Report, University of Washington, 1998.
- [38] O. Zamir and O. Etzioni, Web document clustering: a feasibility demonstration, in: *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, 1998, pp 46-54.

7. Vitae



Oren Etzioni is an Associate Professor in the Department of Computer Science and Engineering at the University of Washington. He received his Ph.D. from Carnegie Mellon University in 1991. After joining the University of Washington he launched the Internet Softbot project. He received an NSF Young Investigator Award in 1993. His research interests include software agents, Web navigation and search technology, and human-computer interaction. See <http://www.cs.washington.edu/homes/etzioni>.



Oren Zamir is a Ph.D. student in the Department of Computer Science and Engineering at the University of Washington. He received his B.S. degree from the Hebrew University, Jerusalem. His research interests include information retrieval, data visualization, machine learning, and knowledge discovery. See <http://www.cs.washington.edu/homes/zamir>.