A Visualization Interface for Document Searching and Browsing

Matthew Carey Frank Kriwaczek Stefan M Rüger

Multimedia Knowledge Management: http://km.doc.ic.ac.uk
Department of Computing, Imperial College
London SW7 2BZ, England

Abstract

We present a text document search engine with several new visualization front-ends that aid navigation through the set of documents returned by a query (*hit documents*). Our methods are based on identifying and selecting keywords on the fly. The choice of keywords depends not only on the frequency of their occurrence within hit documents but also on the specificity of their occurrence just within these hit documents. Keywords are subsequently used to obtain a sparse document representation and to compute document clusters using a variant of the buckshot algorithm. One of the visualization front-ends uses the sparse document representation to obtain keyword clusters.

We make use of the clustering to group the documents returned from the search visually, and to label the groups with their most salient keywords. The different front-ends cater for different user needs. Two of them can be employed to browse cluster information as well as to drill down or up in clusters and refine the search using one or more of the suggested keywords.

CR Categories: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Clustering, Information filtering, Query formulation, Relevance feedback, Selection process; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical user interfaces, Interaction styles

Keywords: Information visualization, document clustering, feature reduction, Sammon map, tree map, radial visualization

1 Introduction

Broad one or two-word searches in conventional search engines are often plagued by low precision, returning thousands of hit documents as their output. A common problem with this is that users may have to sift through much irrelevant material before finding pertinent documents.

We suggest using a standard full-text search engine, but in addition computing keywords from the set of hit-document references. These keywords can assist the user in a variety of ways: informing about issues related to the query, narrowing down the query with additional query term suggestions and clustering, displaying and labelling the hit-document subsets.

In Section 2, we review our method of generating keywords. In order to be computationally efficient we generate a list of candidate keywords for each document at index time. This list is available at query time without having to access the original documents. Three criteria are applied for keyword selection: they must be of general potential interest, be specific for the hit-document set and be of discriminative power within this set. In our document set of around 550,000 documents the query "computer" will produce keywords like "software", "UNIX", "IBM" or "users".

When clustering documents from the hit-document set, we make use of a sparse document representation using keyword histogram vectors. Not only does this alleviate the curse of dimensionality that comes with the otherwise popular word histogram representations but it also reduces the computing time significantly.

Section 3 describes the technical components and interaction of the whole search engine with its Information Navigator front-end, while Section 4 details three graphical interfaces that make use of the keyword-document matrix. These three interfaces are alternative views of the traditional ranked-list representation (see Fig 2).

Our thesis is that these graphical cluster-based representations shift the user's mental load from slower thought-intensive processes such as reading to faster perceptual processes such as pattern recognition in a visual display. In our opinion, the one-dimensional ranked-list metaphor is too restrictive when the hit-documents set is large. Furthermore, in conventional search engines, the documents are ultimately ranked with the aim of ordering them according to relevance to the user. This appears to be overly ambitious as even advanced ranking algorithms cannot know whether the user prefers documents about "hardware" or "software" when the query simply was "computer". Again, a graphical cluster-based interface might help users find what they want.

2 Keywords and Clustering

2.1 The Curse of Dimensionality and Dynamically Computed Keywords

The natural features of text documents are words or phrases, and a document collection can contain millions of such distinct features. The often-used word histogram representation of documents consequently leads to very high dimensional vectors. The intrinsic problem with this kind of representations is that any two randomly picked vectors in a high-dimensional hypercube tend to have a constant distance from each other, no matter what the measure is! As a consequence, clustering algorithms that are based on document distances become unreliable.

To give a numeric example for the effect of high dimensionality, let $x, y \in [0,1]^n$ be drawn independently from a uniform distribution. The expected value of their sum-norm distance is $n/3$, with a standard deviation of $\sqrt{n/18}$. For $n = 1,800$ (corresponding to a joint vocabulary of just 1,800 words for a word histogram representation) this means a typical distance of $|x - y|_1 = 600 \pm 10$. With increasing n, the ratio between standard deviation and vector size gets ever smaller, as it scales with $1/\sqrt{n}$. This is a generic statistical property of high-dimensional spaces with any standard distance measure, and can be traced down to the law of large numbers. For a more detailed discussion about the curse of dimensionality see [19]. Although word histogram document representations are by no means random vectors, each additional dimension tends not only to spread the size of a cluster but also to dilute the distance of two previously well-separated clusters. Hence, it seems prohibitive involving all semantic features (eg, the words) of a document collection for document clustering.

Even after applying standard feature reduction techniques, the number of features remains large. In our clustering experiments with 548,948 mainly US documents[1], a *candidate keyword* had to appear in at least three documents and in no more that 33% of all documents. This resulted in a vocabulary of around 222,872 candidate keywords. In our system we store a set of around 200 candidate keywords per document along with the meta-data of the document, at index time. A set H of documents returned by a query may still have a candidate-keyword vocabulary of well over 10,000 different words. As an example, the four sets of top 500 documents returned by the queries "mafia," "Clinton," "cancer" and "computer" use a vocabulary of between 14,000 and 17,000 different candidate keywords.

Document clustering has attracted much interest in the recent decades, eg [20, 8, 30, 17], and much is known about the importance of feature reduction in general, eg [14] and, in particular, clustering [28]. However, little has been done so far to facilitate feature reduction for document clustering of query results, with the notable exception of [22]. In contrast to the latter paper, which uses a tf-idf weighting scheme, we suggest ranking the importance of each such candidate keyword j with a weight

$$w_j = \frac{h_j}{d_j} \cdot h_j \log(|H|/h_j), \tag{1}$$

where $|H|$ is the total number of hit documents, h_j is the number of documents in H containing word j, and d_j is the number of documents in the whole document collection D containing j. The second factor prefers words with medium-valued matched-document frequencies, while the first factor prefers words that specifically occur in the matched documents. The highest-ranked words are meant to be related to the query. Indeed, we have "software", "IBM", "UNIX" etc as the top-ranked words when querying for "computer". This seems to be a powerful approach to restrict the features of the matched documents to the top k ranked words, which we will call the *keywords*. One important aspect is that the features are computed at query time. Hence, when the above query is refined to "computer hardware", a completely new set of keywords emerges automatically.

2.2 Document Representation and Feature Reduction

For each matched document i we create a k-dimensional vector v_i, whose j-th component v_{ij} is a function of the number of occurrences t_{ij} of the j-th ranked keyword in the document i:

$$v_{ij} = \log_2(1 + t_{ij}) \cdot \log(|D|/d_j) \tag{2}$$

This is a variation of the tf-idf weight, but which gives less stress to the term frequency t_{ij}. We project the vector v_i onto the k-dimensional unit sphere obtaining a normalized vector u_i that represents the document i. We deem the scalar product of u_a and u_b (i.e. the cosine of the angle between vectors v_a and v_b) a sensible *semantic similarity measure* between two documents a and b in the document subset H returned by a query with respect to the complete document collection D.

u may be viewed as a document representation matrix where the row vector u_i is a k-dimensional representation of document i and u_{ij} is viewed as the importance of keyword j for document i. In particular, $u_{ij} = 0$ if and only if document i does not contain keyword j. The number of features k can be controlled by the experimenter, and our experiments using the TREC data of human relevance judgements have shown that $k \approx 10$ yields superior clustering results [19]. Note also that even if only the top ten keywords are used for the clustering and document representation, we might still display more keywords on the screen to assist the user in his or her search.

[1] We were using the two TREC CDs vol4 and vol5 (see http://trec.nist.gov) with articles from the Los Angeles Times, the Financial Times, the Federal Register, Congress Records and the Foreign Broadcast Information Service.

2.3 Document Clustering

Post-retrieval document clustering has been well studied in the recent years, eg [9, 1, 15, 10, 32]. We deploy a variant of the Buckshot algorithm [9], a two-phase process to cluster the document representations u_i in an Euclidean k-dim space. Each cluster contains a certain number of document vectors and is represented by their normalized arithmetic mean, the so-called centroid vector. In the first phase, hierarchical clustering with complete linkage operates on the best-ranked 150 documents (in contrast to the original Buckshot method, which would use a random document sample). This can be done in a fraction of one second CPU time. Hierarchical clustering has the advantage that one can either prescribe the number N of clusters or let the number of clusters be determined by demanding a certain minimum similarity within a cluster. Either way, once clusters within the top-ranked documents are identified, their N centroids can be computed and used as a seed for standard iterative clustering of the remaining documents. This algorithm consumes an amount of time linear in the number $|D|$ of documents and in the number N of clusters. In our experience, one cycle of iterative clustering is not only adequate but also preserves the cluster structure given by those top-ranked documents thought to be the most important. 1,000 documents can thus be clustered in less than one second on a standard 500 MHz PC.

3 System Overview and Architecture

The system can be divided into three major subsystems: 1) the far end components that index the data, process and return the results of the searches, 2) the networking components that transport the data and 3) the display components that present the data and allow the user to interact with it. Fig 1 shows these subsystems in more detail. In the rest of this section we explain the various parts of this architecture.

At the far search engine end there are three major parts to the application, the indexing program that feeds the search engine, the search engine itself and the result processing program that adds the interesting word data, clusters and *Sammon Maps* [21] to the hit-document set.

The networking consists of two parts: a Java threaded socket server that executes the search engine and captures results from it and, secondly, Enterprise Java Beans that communicate with the socket server and return results to a Java servlet that communicates to the display applet.

The display applet consists of a general query interface and three visualizations (in addition to a ranked-list display) on different tabbed panels that are serviced by the same data manager module. This module is the piece of code responsible for storing the data and supplying it to the individual visualization modules.

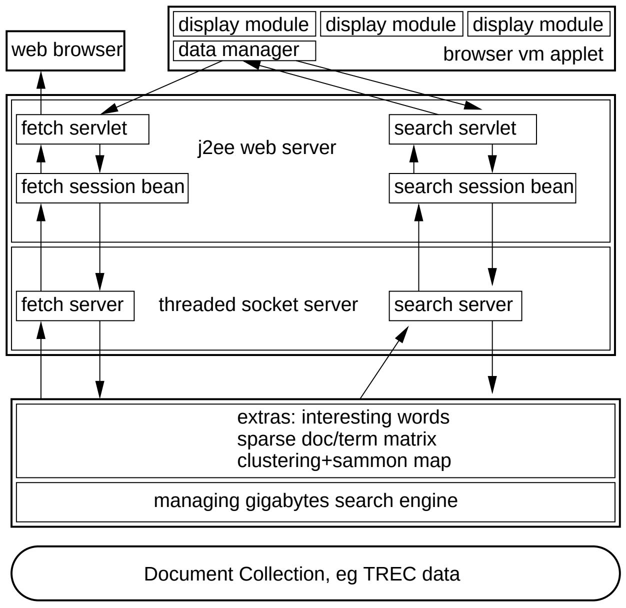


Figure 1: The system architecture

The search engine is powered by a slightly modified version of the *mg search engine* that accompanies the Managing Gigabytes book [31]. This engine, whose C source code is freely downloadable for non-commercial use, can be run in a number of modes. Here we chose to use ranked query searches where the engine generates its reference of the document and a brief section of the text, and we pipe the results out to a processing program.

Before any queries can be run the data set has to be entered into the search engine. The mg engine expects to run an application or a shell script that delivers a series of documents to it via standard output, delimiting each document. Though a basic sample script is provided it is up to the user to devise a suitable script or program to carry out this task. The indexing program fulfils the task of the script. While descending through the various data directories (indicated by an environmental variable) for the documents and listing their contents to the mg engine, it keeps a record of each word used in all the documents and counts how many documents the word occurred in. Having made one complete pass through all the documents the indexing program then has a data file that contains the document frequency of each word in the document data set. This file is used to create an ordered file of candidate keywords, an index into that file and an index file of document frequencies. Candidate keywords are those that fall inside a statistical boundary of document frequencies (see Section 2.1). The indexing program then makes a second pass through the document data files, recording, for each document, the number of occurrences of each candidate keyword in that document. It stores this data in a variable length record file and at the same time creates an index into that file.

Once the documents are indexed, the mg search engine is able to respond to a query by computing efficiently a ranked list of the references to those documents that contain the query words. Note that at this point a standard full-text inverted document file is used for processing a query.

In order to obtain keywords, the candidate keyword and document data encapsulated in the above auxiliary index files are used by the processing program to calculate the weight of each candidate keyword using the statistical formula (1). The 100 highest weighted candidate keywords in the result set are kept as keywords. A sparse matrix of the incidents of the keywords in each document is then created. This matrix is in turn clustered, and a Sammon map is generated from the cluster centroids. The document numbers, document snippets, the keywords, the sparse matrix of keyword document incidents, the clusters of document numbers and the Sammon map of the clusters are all piped out to the calling program.

The threaded socket server is kept continually running, listening for requests coming in on two ports. On a request a thread is created and an instance of the appropriate class to handle the request is also created. In the case of a search query the search server runs a shell script that calls instances of the mg search engine and the result-processing program. When the result is piped back, the search server uses the result data to create a serialized data object. The latter is transmitted back across the network to the search session bean, ending the thread.

An instance of this stateless session bean is continually running within the j2ee server. The bean passes the query on from the servlet to the socket server and then waits for the result, which is passed back to its caller. The servlet also runs within the j2ee server. It gets the query as an http request from the applet running in the browser and sends the query to the stateless session bean.

The applet issues a query and waits for the result. It uses the serialized data that is returned to set the contents of the data manager that is used by all the visualizations. Once this is set up, all the visualizations are notified to recalculate their display.

When the applet issues a request for a document, rather than a query, the request follows a different path. The applet requests the browser to open a page at the url of the fetch servlet, with a request for the document number appended to the url. The fetch servlet then calls the fetch stateless session bean which issues a fetch request to the fetch server. The fetch server within the socket server runs the search engine with a customized request for a document. When the document content is piped back, the server transmits it as a serialized vector of lines to the fetch bean. The fetch bean passes the vector of lines back to the fetch servlet which displays the document in a separate browser window.

The system as it stands is a thick client model. It minimizes use of the network by putting much of the work in the applet. Once the applet is installed in the client's browser only a limited demand will be made on network resources, both at query time and on document retrieval.

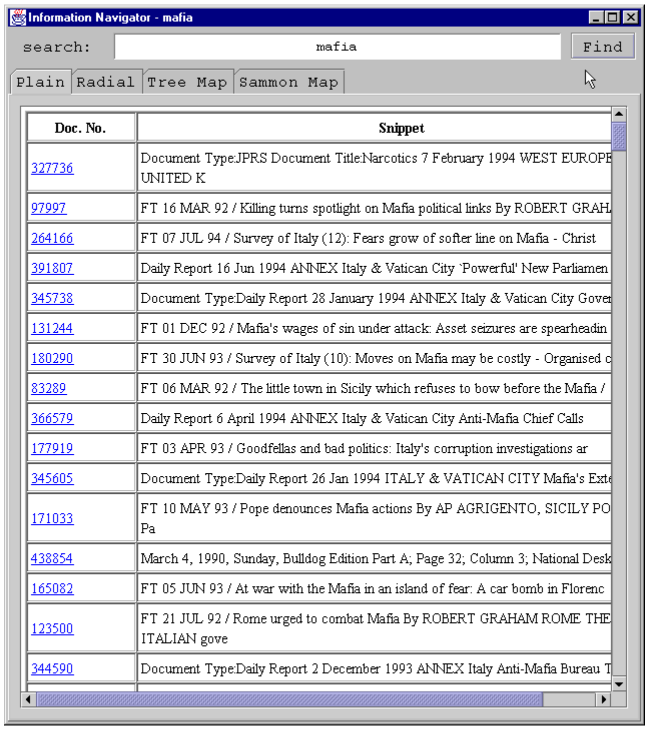
4 New Paradigms in Information Visualization

The last decade has witnessed an explosion in interest in the field of information visualization, e.g. [13, 7, 27, 11, 2, 6, 16, 3, 26, 34, 32, 33, 24, 5, 18]. We add three new techniques for information visualization to the pool of existing visualization paradigms, based on a design study published earlier this year [4].

4.1 Sammon Cluster View

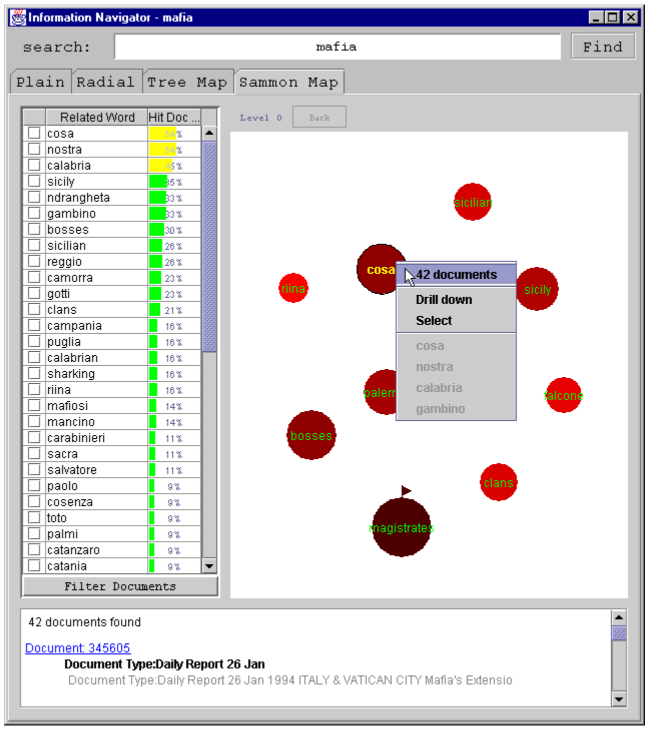
This paradigm uses a Sammon map to generate a two dimensional location from a many-dimensional vector of cluster centroids. This map is computed using an iterative gradient search [21]. The aim of the algorithm is to represent n-dimensional points in a lower, usually 2-dimensional, space while attempting to preserve the pairwise distances between the objects. Hence, a visualization based on the Sammon map attempts to arrange the clusters so that their spatial arrangement in the display rectangle is indicative of their relationship in the n-dimensional space. The idea is to create a visual landscape for navigation.

The display has three panels, a scrolling table panel to the left, a graphic panel to the right and a scrolling text panel below (see



Doc. No.	Snippet
327736	Document Type:JPRS Document Title:Narcotics 7 February 1994 WEST EUROPE UNITED K
97997	FT 16 MAR 92 / Killing turns spotlight on Mafia political links By ROBERT GRAH
264166	FT 07 JUL 94 / Survey of Italy (12): Fears grow of softer line on Mafia - Christ
391807	Daily Report 16 Jun 1994 ANNEX Italy & Vatican City `Powerful' New Parliamen
345738	Document Type:Daily Report 28 January 1994 ANNEX Italy & Vatican City Gover
131244	FT 01 DEC 92 / Mafia's wages of sin under attack: Asset seizures are spearheadin
180290	FT 30 JUN 93 / Survey of Italy (10): Moves on Mafia may be costly - Organised c
83289	FT 06 MAR 92 / The little town in Sicily which refuses to bow before the Mafia /
366579	Daily Report 6 April 1994 ANNEX Italy & Vatican City Anti-Mafia Chief Calls
177919	FT 03 APR 93 / Goodfellas and bad politics: Italy's corruption investigations ar
345605	Document Type:Daily Report 26 Jan 1994 ITALY & VATICAN CITY Mafia's Exte
171033	FT 10 MAY 93 / Pope denounces Mafia actions By AP AGRIGENTO, SICILY PO Pa
438854	March 4, 1990, Sunday, Bulldog Edition Part A; Page 32; Column 3; National Desk
165082	FT 05 JUN 93 / At war with the Mafia in an island of fear: A car bomb in Florenc
123500	FT 21 JUL 92 / Rome urged to combat Mafia By ROBERT GRAHAM ROME THE ITALIAN gove
344590	Document Type:Daily Report 2 December 1993 ANNEX Italy Anti-Mafia Bureau T

Figure 2: The traditional ranked-list metaphor



	Related Word	Hit Doc ...
☐	cosa	54%
☐	nostra	54%
☐	calabria	45%
☐	sicily	35%
☐	ndrangheta	33%
☐	gambino	33%
☐	bosses	30%
☐	sicilian	26%
☐	reggio	26%
☐	camorra	23%
☐	gotti	23%
☐	clans	21%
☐	campania	16%
☐	puglia	16%
☐	calabrian	16%
☐	sharking	16%
☐	riina	16%
☐	mafiosi	14%
☐	mancino	14%
☐	carabinieri	11%
☐	sacra	11%
☐	salvatore	11%
☐	paolo	9%
☐	cosenza	9%
☐	toto	9%
☐	palmi	9%
☐	catanzaro	9%
☐	catania	9%

42 documents found

Document: 345605

Document Type:Daily Report 26 Jan

Document Type:Daily Report 26 Jan 1994 ITALY & VATICAN CITY Mafia's Extensio

Figure 3: Sammon-mapping for clustering-based search

Fig 3). In the graphic panel each cluster is represented by a circle and is labelled with its most frequent keyword. The larger circles are darker and represent the larger clusters, the smaller circles represent smaller clusters and are a brighter red colour. The distance between any two circles in the graphic panel is an indication of the similarity of their respective clusters: the nearer the clusters, the more likely the documents contained with in will be similar. When the mouse passes over the cluster circle a 'tool tip' box in the form of a pop-up menu appears. Operations such as "select" and "drill down" can be executed for this particular cluster or a selection of clusters.

The first item in the cluster popup menu shows the count of documents in that cluster. Choosing this item displays a scrolled table of cluster keywords in the pane on the left-hand side of the visualization and a scrolled list of cluster document hot-links and snippets appear in the scrolling text window at the bottom of the display.

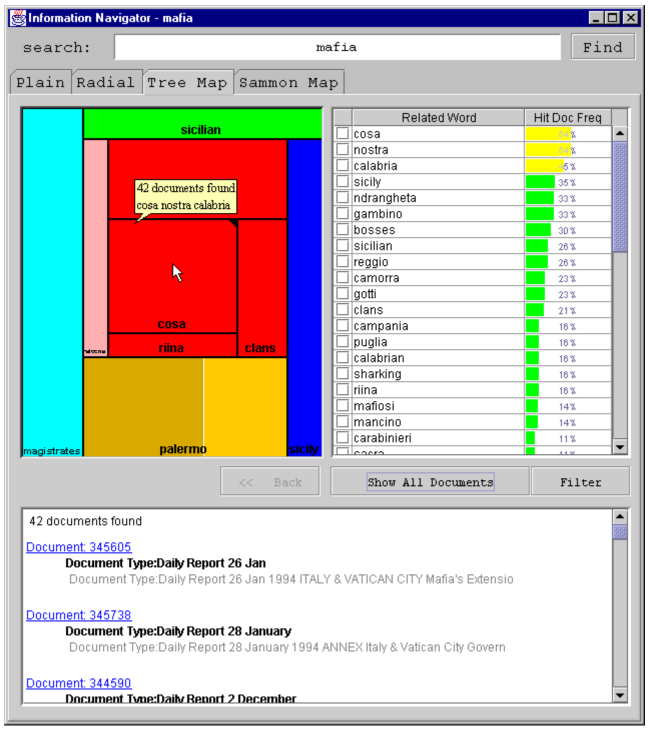
The table of keywords includes a box field that can be selected. At the bottom of the table is a filter button that makes the scrolling text window display only the hot-links and snippets from documents that contain the selected keywords. The select item in the pop-up menu marks a cluster as selected and signals this with a flag. The other menu items serve to label the cluster with four significant keywords and are not selectable.

The drill down item in the pop-up menu performs a redisplay of the documents in the current cluster and all selected clusters (if any). Drill down in this sense pushes the current data manager onto a stack and creates a new data manager that consists of only the documents in the current and selected clusters. This new instance of the data manager re-clusters the subset of the original hit-document set and then creates a new Sammon map that in turn leads to a new display in this visualization. The level indication at the top of the display is incremented and the back button enabled. The back button pops the data manager from the stack and climbs up the hierarchy (drill up). The clustering algorithm used for reclustering on a drill down operation is essentially the same as the one described in 2.3. This time though it is implemented in Java within the applet (as opposed to C in the server processing code). As long as there are more that six documents in the selected set of documents the drill down operation is enabled.

4.2 Tree-Map Visualization

The Tree Map Visualization represents the clusters of documents as rectangles (see Fig 4). This representation is based on an idea found in [25]. The rectangles are arranged so that they fit within a larger rectangle. The size of each rectangle is proportionate to the size of the cluster it represents. Similar clusters are grouped together in *super clusters*. Super clusters are separated by black lines, while sub clusters have white lines between them. Cluster rectangles are different colours, and are labelled with their most frequent keywords. The algorithm used to arrange and size the rectangles is that described in [25]. Moving the mouse over a cluster rectangle brings up a bubble box illustrating the three most numerous keywords in the cluster. Clicking with the left mouse button on a rectangle displays the list of keywords that occur in the cluster in a scrolling table on the right of the display. The keywords are listed in descending order of cluster document frequency. A bar indicates the frequency of each keyword accompanied by the percentage amount. Keywords can be selected as in the Sammon map view. The 'Show all Documents' button below the table lists the snippet and hot-linked document number for every document in the cluster in the scrolling box at the bottom of the frame. Clicking on the hot-link displays the document in a separate document window. The filter button, when clicked, displays just the documents that contain the keywords selected in the table pane, in the box at the bottom of the frame. Right-clicking on the mouse over a cluster rectangle causes a drill down to occur.

Drill down, as implemented in this visualization, is a new search of the archive document set using the most populous keywords as the search terms. The visualization pushes its record of the data



search: mafia

42 documents found
cosa nostra calabria

	Related Word	Hit Doc Freq
☐	cosa	54%
☐	nostra	54%
☐	calabria	45%
☐	sicily	35%
☐	ndrangheta	33%
☐	gambino	33%
☐	bosses	30%
☐	sicilian	26%
☐	reggio	26%
☐	camorra	23%
☐	gotti	23%
☐	clans	21%
☐	campania	16%
☐	puglia	16%
☐	calabrian	16%
☐	sharking	16%
☐	riina	16%
☐	mafiosi	14%
☐	mancino	14%
☐	carabinieri	11%

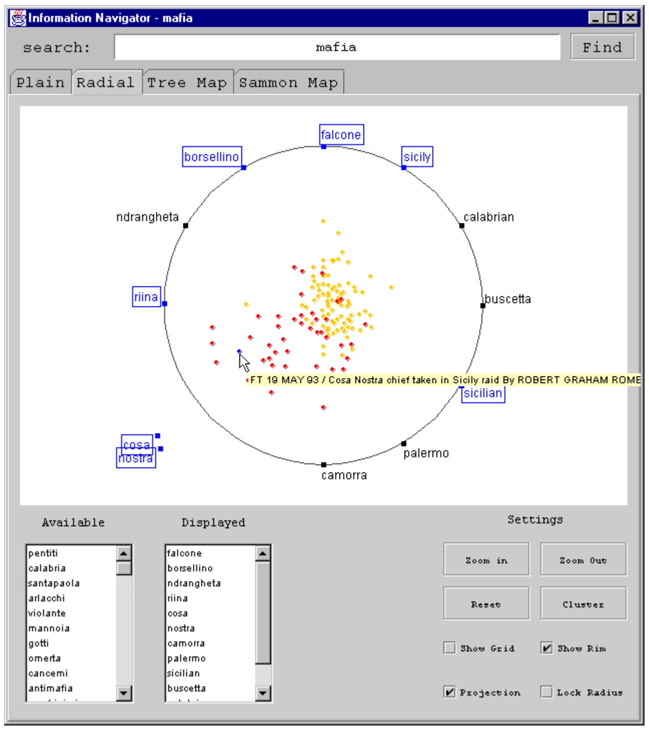
42 documents found

Document: 345605
Document Type:Daily Report 26 Jan
Document Type:Daily Report 26 Jan 1994 ITALY & VATICAN CITY Mafia's Extensio

Document: 345738
Document Type:Daily Report 28 January
Document Type:Daily Report 28 January 1994 ANNEX Italy & Vatican City Govern

Document: 344590
Document Type:Daily Report 2 December

Figure 4: Tree-Map visualization



search: mafia

FT 19 MAY 93 / Cosa Nostra chief taken in Sicily raid By ROBERT GRAHAM ROME

Available	Displayed
pentiti	falcone
calabria	borsellino
santapaola	ndrangheta
arlacchi	riina
violante	cosa
mannoia	nostra
gotti	camorra
omerta	palermo
cancemi	sicilian
antimafia	buscetta

☐ Show Grid [✓ Show Rim]
[✓ Projection] ☐ Lock Radius

Figure 5: Radial visualization

manager onto a stack. It then creates a new data manager that executes the query request and returns the results to this visualization. When the result is returned a little animation illustrates drill down and a new display is created within a slightly smaller rectangle. The back button becomes enabled. If clicked, the display moves back to the previous visualization. Note that this is a different interpretation of 'drill down' to that used in the Sammon Map visualization.

4.3 Radial Interactive Visualization

Radial is a visualization that is very similar to VIBE [13], to Radviz [12] and to Lyberworld [11]. Radviz places the keyword nodes round a circle as dimensional anchors and the document nodes occur in the middle as if suspended by springs connecting them to keyword nodes. The greater the content, the stronger the springs affect on the document's location. (Fig 5 illustrates this paradigm.) Hence, we make direct use of the representation matrix u without explicit clustering. Radial adds a level of user interaction to the metaphor introduced by RadViz and VIBE. Building on VIBE, Lyberworld takes a similar idea into three dimensions. In Lyberworld vector addition is used to position the documents nodes within a *relevance sphere*. The document's keyword content creates a vector between the centre axis of the sphere and the position of that keyword on the sphere's surface. The radius of the sphere is defined by the range of possible vector lengths. In Lyberworld the relevance sphere can be manipulated, rotated so that the 2D computer display can give more of a clue as to the 3D location of the document node relative to the keyword nodes. Also the relative attractiveness of the keyword nodes can be enhanced to pull related documents towards them. Radial, staying with only two dimensions, dispenses with some of the perceptual complexity implicit in rendering a three dimensional model on a two dimensional screen.

Radial, as in all the visualizations we present here, uses the statistically collected keywords to differentiate the documents. Initially, the twelve highest ranking keywords are displayed in a circle. Any documents in the search set that contain those keywords are placed using a similar vector sum within the circle. As the mouse passes over the document nodes, a bubble displays a descriptive piece of text from the document. The dimensions of the circle are more arbitrary than in Lyberworld and if the display were simply based on a flat sum of vectors it would be possible for the document nodes to be outside the circle. However we have constrained their positions by projecting the radial locations through the arc-tangent function. Thus it becomes harder and harder for a document to be moved towards the edge of the circle.

Radial, like VIBE, RadViz and Lyberworld, attempts to present a great many dimensions in only few. Thus locations of document nodes can be ambiguous, and there is more than one possible reason why a node could be at a particular point. To mitigate this in Radial the user can click on a document node, and the keyword nodes that affect the location of document are highlighted. We believe that this is a novel and useful feature. Clues to the effects of different dimensions are also given when a keyword node is selected with the mouse: the document nodes which contain that keyword are highlighted.

Similarly to Lyberworld the vector effect of a particular keyword node on the document set can be enhanced. However, in Radial it is achieved by grabbing the keyword node with the mouse and moving it away from the edge of the circle. A sort of manual clustering can be effected placing several keyword nodes together (like the "cosa" and "nostra" keywords in Fig 5). Alternatively, a button allows the displayed keyword arrangement to be automatically clustered using the columns of the matrix u. Note that document clustering was done using the *rows* of this matrix.

The choice of keywords used in the display can be enhanced and reduced by clicking on two visible lists of words. Zoom buttons allow the degree of projection to be increased or reduced so as to distinguish between document nodes around the edges of the display or at the centre.

Double clicking on a document node with the mouse retrieves the full document in a separate browser window.

The shortcoming of the Radial visualization is that it can only say something about the documents in the result set that contain the particular, limited set of keywords which are displayed. When too many keywords are displayed the whole display becomes difficult to interpret. This is where the cluster-based visualizations as witnessed here in the Tree Map visualization and the Sammon Map visualization show their strength. On the other hand, the Radial visualization appears to be a good interactive tool to structure the document set according to one's own preferences by shifting keywords around in the display.

4.4 Unified Approach

The unified approach brings the possibility that different visualizations can be plugged in and that the different visualizations used could be compared and evaluated against one another.

The application as a whole offers the possibility of browsing the same result set in several different ways simultaneously, flipping from one visualization to another. The cluster-based visualizations give a broader overall picture of the result, while the Radial visualization allows the user to focus on subsets of keywords. Also, as the clusters are, by implication, approximations that bring to the fore particular keywords, it may be useful to return to the Radial visualization and examine the effect of these keywords upon the whole document set. The Sammon Map attempts to say something more about the relative relationship of the clusters to one another than the Tree Map, while the Tree Map is more explicit about cluster size and shows a two-level hierarchical cluster structure.

The Radial visualization will perhaps be more fruitful if the initial keywords match the users area of interest. However the Tree map visualization will allow the user to focus on the keywords that inhabit the sort of documents the user is interested in, even allowing the user to formulate a perhaps more productive search. The Sammon Map will let the user dissect the search set and re-cluster a subset, to gradually move closer to the sort of documents that contain the items the user is interested in. Again the user may have recourse to the Radial visualization and apply those keywords to the whole result set. In the end, he or she will need to read some of the documents, but not as many as by negotiating through a ranked list.

A user wishing to browse a well-structured system of documents (like a library) is using a system that has, in a sense, already been clustered. Like documents are obviously arranged close to one another. The cluster-based visualizations give a visual analogy of the structure implicit in the library classification scheme. The Radial visualization throws up the effects of keywords that cause cross references between documents, and allows the user to skim between subject areas. Likewise, the drill down as implemented in the Tree Map Visualization reformulates the search terms, perhaps taking the user into different classification areas.

5 Evaluation

The literature of information visualization is still one mainly of construction, not of evaluation; for some exceptions to this rule see [29, 23]. Our evaluation is a three level approach. For level one, we performed experiments to assess the quality of the clustering process based on human-expert data, ie, the ability to separate relevant documents from irrelevant documents [19]. We used the 1997-1998 subcollection of the TREC data with 528,155 documents, 100 queries and corresponding relevance assessments. The question we posed was whether our clustering algorithms would produce clusters in which the concentration of relevant documents was either very high or very low. The results showed a compelling evidence for the validity of the Clustering Hypothesis [28] for post-retrieval document sets and for the use of low-dimensional document representations using the keyword computation discussed in Section 2. This study gave the green light for developing designs for cluster visualizations.

For level two, in the process of developing this software, we were able to get feedback from acquaintances and students through questionnaires. This was very useful. Placing the executable applet on the web and letting people know about it meant that shortcomings were made known to us very quickly.

The application was posted on the web. We asked a group of users to complete a pre-use questionnaire. We then had them employ specific visualization interfaces to execute a collection of preset queries that were sufficiently obscure for the answers not to be known beforehand, and yet to be in the data set somewhere. Next, we asked the users to apply the system to queries of their own choice. By this point they would very likely have definite views on the visualization interfaces. Finally, we asked them to complete a post-use questionnaire that contained both set, narrow, fixed questions about the application and more open areas for user comment. This second-level evaluation served to refine the design and development of the visualization interfaces.

The third level of evaluation is a proper and formal user study and scheduled to be carried out in November 2000 through collaboration with the Ben-Gurion University of the Negev.

6 Conclusions

Our work has contributed to the visualization and browsing of the set of document returned by a search engine in a number of ways. 1) We can identify relevant features of this document set: the keywords. These are used for dimensionality reduction and improved clustering, cluster labelling, query refinement and visualization. 2) Using Sammon's algorithm we are able to create a setting with a holistic view giving primarily information about a first-order cluster structure and inter-cluster relations. The main purpose is to quickly weed out irrelevant clusters and drill down in one or more relevant clusters. 3) Using the Tree-Map algorithm, we are able to display second-order cluster structure at one glance. Applications include learning about the fine-structure and nature of queried object as coded in the actual use of the keywords in the document repository - ideal for a user knowing little about the subject. 4) A keyword clustering with the Radial visualization gives rise to another novel document clustering approach, one where the user can control the building of groups by interactively moving the keywords around. We feel that this interface is particularly useful for an experimental, user-driven approach to form clusters and to get a suitable ranking by interactively moving the keywords around on the screen.

References

[1] R B Allen, P Obry, and M Littman. An interface for navigating clustered document sets returned by queries. In *Proc of the ACM Conf on Organizational Computing Systems*, pages 166–171, 1993.

[2] M Ankerst, D Keim, and H Kriegel. Circle segments: A technique for visually exploring large multidimensional data sets. In *IEEE Visualization '96*, 1996.

[3] J Assa, D Cohen-Or, and T Milo. Displaying data in multidimensional relevance space with 2d visualization maps. In *IEEE Visualization '97*, 1997.

[4] P Au, M Carey, S Sewraz, Y Guo, and S M Rüger. New paradigms in information visualisation. In *Proc of the 23rd International ACM SIGIR Conference*, 2000.

[5] K Borner. Visible threads: A smart vr interface to digital libraries. In *Proc of IST/SPIE's 12th Annual International Symposium: Electronic Imaging 2000: Visual Data Exploration and Analysis (SPIE 2000)*, 2000.

[6] S K Card. Visualizing retrieved information: A survey. *IEEE Computer Graphics and Applications*, 16(2):63–67, 1996.

[7] M Chalmers and P Chitson. Bead: Explorations in information visualisation. In *Proc of the 15th Intl ACM SIGIR Conf*, 1992.

[8] W B Croft. *Organizing and searching large files of documents*. PhD thesis, University of Cambridge, October 1978.

[9] 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 *Proc of the 15th Intl ACM SIGIR Conf*, pages 318–329, 1992.

[10] 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*, 1996.

[11] M Hemmje, C Kunkel, and A Willet. Lyberworld - a visualization user interface supporting fulltext retrieval. In *Proc of the 17th Intl ACM SIGIR Conf*, 1994.

[12] P Hoffman, G Grinstein, and D Pinkney. Dimensional anchors: A graphic primitive for multidimensional multivariate information visualizations. In *Proc of the NPIV 99*, pages 9–16, 2000.

[13] R Korfhage. To see or not to see - is that the query? In *Proc of the 14th Intl ACM SIGIR Conf*, 1991.

[14] P R Krishnaiah and L N Kanal, editors. *Handbook of Statistics: Classification, Pattern Recognition and Reduction of Dimensionality*, volume 2. North-Holland Publishing Company, 1982.

[15] 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.

[16] L T Nowell, R K France, D Hix, L S Heath, and E A Fox. Visualizing search results: Some alternatives to query-document similarity. In *Proc of 19th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR '96)*, 1996.

[17] E Rasmussen. Clustering algorithms. In W B Frakes and R Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice Hall, 1992.

[18] P Rheingans and M desJardins. Visualizing high-dimensional predictive model quality. In *Proc of IEEE Visualization 2000*, 2000.

[19] S M Rüger and S E Gauch. Feature reduction for document clustering and classification. Technical report, Computing Department, Imperial College, London, UK, 2000.

[20] Gerard Salton. *Automatic information organization and retrieval*. McGraw-Hill, New York, 1968.

[21] J W Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5), 1969.

[22] H Schütze and C Silverstein. Projections for efficient document clustering. In *Proceedings of the 20th International ACM SIGIR Conference*, 1997.

[23] M Sebrechts, J Vasilakis, M Miller, J Cugini, and S Laskowski. Visualization of search results: A comparative evaluation of text, 2d and 3d interfaces. In *Proc of 22nd International ACM Conference on Research and Development in Information Retrieval (SIGIR '99)*, 1999.

[24] C D Shaw, J M Kukla, I Soboroff, D S Ebert, C K Nicholas, A Zwa, E L Miller, and D A Roberts. Interactive volumetric information visualization for document corpus management. *International Journal on Digital Libraries*, 2(2/3):144–156, 1999.

[25] B Shneiderman. Tree visualization with Tree-Maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[26] B Shneiderman, D Feldman, A Rose, and X Ferre' Grau. Visualizing digital library search results with categorical and hierarchical axes. In *Proc of ACM Digital Libraries 2000*, 2000.

[27] A Spoerry. Infocrystal: A visual tool for information retrieval & management. In *Proc of Information, Knowledge and Management 93*, 1993.

[28] C J van Rijsbergen. *Information Retrieval*. Butterworth, London, 2nd edition, 1979.

[29] A Veerasamy and R Heikes. Effectiveness of a graphical display of retrieval results. In *Proc of 20th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR '97)*, pages 85–92, 1997.

[30] E Voorhees. The cluster hypothesis revisited. In *Proc of ACM SIGIR*, pages 188–196, 1985.

[31] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes*. Morgan Kaufmann Publishers, 1999.

[32] O Zamir and O Etzioni. Web document clustering: A feasibility demonstration. In *Proc of the 21th Intl ACM SIGIR Conf*, pages 46–54, 1998.

[33] O Zamir and O Etzioni. Grouper: A dynamic clustering interface to web search results. In *Proc of the Eighth International World Wide Web Conference (WWW8)*, 1999.

[34] M Zhou and S Feiner. Visual task characterization for automated visual discourse synthesis. In *Proc of Conference on Human Factors in Computing Systems (CHI'98)*, pages 392–399, 1998.

Acknowledgements: This work was partially supported by the EPSRC, UK.