Available online at www.sciencedirect.com
SCIENCE DIRECT®
Decision Support Systems 37 (2004) 415–434
Decision Support Systems
www.elsevier.com/locate/dsw

Development of a fisheye-based information search processing aid (FISPA) for managing information overload in the web environment

Ozgur Turetken[a,*], Ramesh Sharda[b]

[a] Fox School of Business and Management, Temple University, Philadelphia, PA 19122, USA
[b] College of Business Administration, Oklahoma State University, Stillwater, OK 74078, USA

Received 1 June 2002; accepted 1 October 2002

Available online 10 April 2003

Abstract

Information technologies have proliferated at an unprecedented rate to provide access to information across geographical boundaries. However, this proliferation has led to an information overload. Information overload has adverse impacts on information use and decision quality. This research focuses on the overload problem resulting from a web search, and proposes a potential remedy. We develop the requirements of a system that makes use of clustering and visualization for browsing the results of a typical web search. Based on this model, we develop a prototype that visualizes search results by first organizing them into a hierarchy according to their individual contents. This system presents a visual overview of the groups in this hierarchy, and lets the users focus (zoom) on specific groups of interest. One general problem with zooming within hierarchical structures is the separation between the details and the context. To address this problem, we implement a fisheye zooming capability in our system. This paper describes a typology of the various components necessary for addressing the problem and then the proposed solution based upon a fisheye view-based visualization. Next, the specific visualization algorithm and the system implementation are described. We conclude with research questions for further development of such interfaces for presentation of the results from web searches.

Keywords: Information visualization; Web search; Human–computer interaction; User-interface design; Knowledge management

1. Introduction

One of the biggest challenges for successful management of information in today's ''information-intensive'' world is to find efficient ways to reduce information overload. Information overload occurs when an information user is exposed to more information than she needs and, more importantly, is able to process. This problem has long been recognized and studied in various disciplines such as accounting [5] and finance [36] among others [32]. However, the challenge is much more serious today because the almost ubiquitous Internet makes it extremely easy to access various kinds of information created virtually

* Corresponding author.
E-mail addresses: turetken@temple.edu (O. Turetken), sharda@okstate.edu (R. Sharda).

0167-9236/03/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0167-9236(03)00047-2

anywhere. While this phenomenon offers unprecedented opportunities for the sharing of information, it also creates massive amounts of overload. Information overload poses serious threats to the quality of information that one uses and bases their decisions upon, as people tend to somewhat arbitrarily filter information when they are overloaded [18,26].

In this study, we address the information overload problem generated by a web search. Typically, web search engines present results as a ranked list. For broadly formulated search queries, such a list may contain thousands of documents. Research has suggested that search engine users are not likely to go beyond the top 20 to 30 documents on these lists before they get bored or frustrated, and subsequently quit the search [31]. Web search success is thus degraded. One remedy to this problem is finding more efficient ways of filtering information (e.g. Ref. [8]) to pursue higher precision in search results. Filtering is useful in increasing precision, especially for relatively small-scale repositories. However, filtering the irrelevant information from the search results increases the ratio of the relevant information to the irrelevant, but does not increase the chance of retrieving more of the potentially relevant information. This issue is addressed by research on web indexing and search algorithms [3,10], as well as query formulation [11,35]. As a result of these efforts, web search is more effective and easier today than it was in the earlier days of the Internet. Yet, the overload problem in web search is still very evident. Accordingly, additional support in processing the results of a web search is valuable. This paper reports on our research efforts in providing such post-search support at the user-interface level.

To enhance the post-search exploration process, we designed and implemented a prototype system (called fisheye-based information search processing aid—FISPA) that is based on information visualization. Information visualization is the common name for the techniques that use the idea of supporting the cognitive system by means of visual cues [39,41]. The relative processing capacity and speed advantage of the perceptual (visual) system to the cognitive system results in the better and quicker understanding of information when supported by visual cues. Some well-known applications of this idea in our domain of interest are the use of visual maps to represent directory structures [4,17,37,38] and visual representation of documents returned as a result of a database query [14].

Visualization of web content has been of research interest since the early days of the World Wide Web [2,13,15,23,27,29]. Most of the work on web visualization focuses on visualizing the link structure of a web site. Because the list provided as a result of the search process does not have an inherent structure, the presentation of such a collection of web pages requires a high level of organization before the visualization process. Earlier work on the presentation of results to a traditional database query has utilized clustering (grouping) to provide such organization [15].

Clustering is commonly used to identify patterns in an unstructured collection of objects. When a collection is examined through a cluster structure, it is easier to discover quick facts about it and to move on to the examination of individual objects. Clustering has its use in various application domains such as market segmentation in traditional market research, and in various forms of data mining. Accordingly, we adopted a clustering-based visualization in our design.

One challenge in applying clustering to the visualization of web search results is finding appropriate representations for web documents to be able to cluster them [31,45]. Recent work in web visualization [31] has approached the problem by preprocessing web documents to extract their textual contents. Once this is done, techniques of representing and clustering text documents that have been around long before the invention of the Internet [33] can be applied to these documents.

The use of clustering in the organization of web search results creates a multilevel information structure. One major challenge in visualization of such structures is to facilitate the user's perceptual integration of the separate views of the context and details. More often than not, user interfaces to hypermedia systems fail to meet this challenge, which leaves users feeling disoriented or "lost in hyperspace" [1,29]. The fisheye view idea [12], based on providing views of details embedded in context, can be used to address this issue since such views provide a smooth integration of context and details. In our design, we employ a version of fisheye views in the

visualization of web search result clusters. The fisheye view approach allows for zoom to detail while keeping a global context in view. Our system provides an overview of clusters by means of a two-dimensional map, in which details are examined by zooming on the regions of more interest. We implemented two different kinds of zooming: a fisheye zoom that presents details in context, and the traditional method of full zoom, which strictly separates the views of different levels of the hierarchy, that is, context and details.

This paper has two major objectives. The first objective is to develop the design requirements for a system that is based on the combined use of clustering and visualization for the reduction of information overload in exploration of web search results. This design includes the concept of fisheye views to facilitate more successful browsing of web search results once they are retrieved. The second objective is to introduce a visualization algorithm, and integrate this algorithm with the best available tools to implement the specified design.

The paper is organized as follows. The next section proposes a system architecture for a clustering-based visual web search processing aid accompanied by a brief review of related previous work. It also includes a background on fisheye views. Next, the algorithms used in the design of the system and implementation of the prototype are described, followed by a description of the prototype implementation. A sample session with our system, and the results of the preliminary usability studies are illustrated then. We conclude with plans for enhancement of the system in Section 6.

2. A system architecture for a clustering-based visual web search processing aid

Based on the previous research briefly mentioned in the previous section, we identify three general tasks that need to be performed for the visual presentation of web search results (see Fig. 1). These tasks comprise an overall set of capabilities a post-search processing system should include. This section provides a brief discussion of these capabilities along with representative examples of related previous work.

The overall set of functions such a system should provide could be summarized as follows. The search processing aid should take the results of a web search and create a multidimensional vector, which repre-

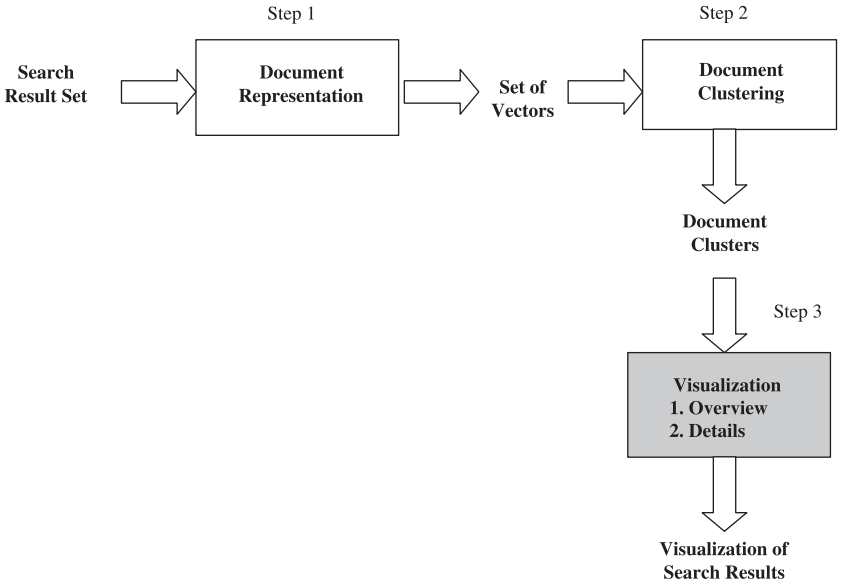

Fig. 1. The model for a clustering-based visual search aid.

sents this collection. This vector is then made available to the clustering step so that the resultant documents are organized into a single or multilevel cluster. Through the visualization process, the reader can explore the clusters and the documents. The visual display system allows zooming to any individual document or a cluster with or without a fisheye zoom capability. Stated formally:

Given a search result set containing n documents $\Re_n$:

$$I_m = f_1(\Re_n) \tag{1}$$

where I_m is a set of m index terms extracted from $\Re_n$

$$\mathfrak{U}_{nm} = f_2(\Re_n, I_m) \tag{2}$$

where $\mathfrak{U}_{nm}$ is a set of m-dimensional vectors one corresponding to each document in $\Re_n$

$$C = f_3(\mathfrak{U}_{nm}) \tag{3}$$

where C is a cluster structure of the collection, and

$$\text{Map}_{\text{full}} = f_4(C) \text{ and } \text{Map}_{\text{fisheye}} = f_4'(C) \tag{4}$$

where Map_{full} and $\text{Map}_{\text{fisheye}}$ are the full-zoom or fisheye visualizations of the cluster structure C. The objective is to develop and implement specific algorithms for f_1, f_2, f_3, f_4, or f_4'.

This model is general enough to be a framework for design of similar systems. Contributions to the development of a general web search aid could be made in any step of the model by developing an appropriate implementation of f_1, f_2, f_3, f_4, or f_4'. Of course, we recognize that considerable research continues to pursue improved indexing and search algorithms. Our focus, however, is on processing a set of results $\Re_n$ produced through a search. The following three subsections describe the functionality of such systems in general while also pointing to research in these areas.

2.1. Document representation

Processing the results of a web search requires that these documents be represented in a form amenable to further processing. A common approach to model a document is based on its semantic contents. With this approach, documents are represented as vectors where each element in a vector corresponds to a term in the specific collection of documents [33]. An element in a vector represents the weight of the corresponding term in the specific document that the vector represents. The calculation of weights is performed on a list of terms in each document, which remain after the elimination of noise words, those that are used frequently in the English language, but do not carry unique meanings in each document, such as "is," "the," "a," "of," "but," and "an."

Tkach [40], among many others, observed that using single words for extracting representative terms in a document collection is inefficient. A more efficient alternative is the use of lexical affinities. "A lexical affinity is a correlated group of words, which appear frequently within a short distance of one another. Lexical affinities include phrases like 'online library' or 'computer hardware' as well as other less readable word groupings. They are generated dynamically. Thus, they are specific for each collection. A set of semantically rich terms can be obtained without a need to hand-code a specialized lexicon or a thesaurus" [40]. Accordingly, many document-indexing systems represent documents using a combination of lexical affinities and individual key words instead of words only [31,33].

2.2. Clustering

Once documents are represented mathematically as vectors, a clustering algorithm can be applied to this collection. There are two main types of clustering algorithms that differ according to the final groupings they create. Nonhierarchical techniques divide (partition) a data set into a series of subsets, which are comprised of similar objects with no hierarchical relationship between them. The cluster structure resulting from a nonhierarchical technique is dependent on a number of predefined parameters such as the desired number of clusters, and therefore tends not to be very stable [43].

A hierarchical cluster scheme is composed of smaller clusters within larger clusters. The largest cluster is the whole collection of objects being grouped. The result is a tree-like structure, with individual objects residing on the leaves and larger clusters being reached as one climbs higher in the tree toward its root. A very well-known example of this type of collection is the directory structure on a computer's hard disk. In

such a structure, there are a number of directories (starting from the root) within which files and/or other directories are stored. Hierarchical clustering algorithms are general enough to be applicable to non-hierarchical clustering with a few modifications.

The applicability of clustering to presenting an unstructured collection of documents has been commonly recognized. The Scatter/Gather system [6] is one such clustering-based system that is directed toward a focus set of documents potentially interesting to the user. The focus set is clustered into smaller subsets and summarized to form an outline from which the user can select a smaller focus set. The indicated subcollection becomes the focus set, and the process repeats [7]. Pirolli et al. [28] tested the effectiveness of Scatter/Gather as a simple document retrieval tool, and studied its effects on the incidental learning of topic structure. Their basic conclusion was that the Scatter/Gather clustering method improved browsing effectiveness. Hearst and Pederson [15] revisited the method, and applied Scatter/Gather to retrieval results concluding that clustering increased both search effectiveness and efficiency. The results of the Scatter/Gather studies provide strong evidence to the successful use of clustering in information retrieval.

2.3. Visualization

In our model, the approach to presenting search results is to not only cluster them, but to also present visual representations of these clusters. Visual systems provide such representations, that is, overviews of an information space, and provide zooming capabilities for the users to interact with these overviews as well as focus on specific areas of interest. Graphs are the general data structures used in representing similarities within a group of objects such as documents and document clusters. Accordingly, visualization of a cluster structure can be studied as a specific paradigm in the area of graph visualization. In their comprehensive survey of graph visualization, Herman et al. [16] describe a number of graph drawing algorithms, most of which are directly applicable to the web domain. Numerous visual metaphors such as a book [2], newspaper [13], and even a city [9] can be used to represent a graph, yet the majority of reported systems in the visualization literature use a tree or a map as their visual metaphor. Hence, we examine these metaphors in more detail.

A two-dimensional tree is the most common presentation of a hierarchy. For example, it is very typical to visualize an organizational chart by means of a two-dimensional tree. Subsequently, two-dimensional tree visualizations have been used in interfaces of computerized systems. The Pruning with Dynamic Queries (PDQ) Tree-browser [20] is such an interface, which provides an overview of a hierarchy (including that of a web site) by means of a two-dimensional tree where areas of more interest (i.e. branches) are zoomed as needed. The system was designed to help information users browse hierarchies in searching for the nodes of most interest to them. The purpose of pruning is to reduce the set of alternatives when the viewer is trying to decide which branch to choose to find the information she is seeking. The drawback of the PDQ system is that it does not provide strong visual cues to smoothly connect details and context.

Although not as common as their two-dimensional counterparts, three-dimensional trees have the advantage of being able to display a larger information space than a two-dimensional tree could. Like two-dimensional trees, three-dimensional trees are used to visualize hierarchies where certain branches of the tree can be brought into focus. One of the best known examples of three-dimensional trees comes from Xerox PARC and is known as the Cone tree [30]. In a cone tree, branches of interest are focused by rotating a portion of the tree. The problem with this approach is the occlusion of certain parts of the tree by the focused branch. Another problem is that it is difficult to understand the transition between the overview and details after zooming. Tree visualizations are very intuitive, and for that reason, designers are likely to continue using them. Yet, due to some of the drawbacks mentioned above, there have been attempts to use different visual metaphors to represent hierarchies. Maps are examples of such metaphors.

Geographic maps have a long history of providing spatial clues by means of an overview of a geographical area of interest. A map can similarly be used to present an overview of an information space, with different regions representing different groups (clusters). The proximity of regions means that the underlying concepts have close semantic contents

while the size of a region is an indicator of the size of the corresponding cluster. This idea has been extensively used in information retrieval interfaces. Some common examples of two-dimensional maps for the web are WebSOM from the Helsinki University of Technology [21], the CategoryMap from the University of Arizona [4], the Visual Site map from the University of Kentucky [25], the Galaxies visualization in SPIRE [44], and Cartia's Themescape. WebSOM, CategoryMap, and Visual Site are based on Kohonen's [19] self-organizing maps, an artificial intelligence technique motivated by the clustering capabilities of the human brain. In a recent application, Roussinov and Chen [31] used a variant of this approach to create an overview of results of a typical web search. The basic idea in this application was to take the clustering approach one step further, and make the clusters visible. The authors empirically showed that this approach increased search speed, and it was preferred by most of its users.

A common feature among the visual systems reviewed above is that they give undistorted views of an information space. Two main classes of such systems exist, and are characterized according to the zooming capabilities they provide. The first class of such systems displays the zoomed-in area of the visualization in full detail and prunes the area that is not in the zoom (strict filtering). Others provide separate views of the context and details. Alternatively, to accomplish smooth integration of context and details as we aim in our design, a distortion-oriented approach can be adopted. A majority of distortion-oriented systems are based on Furnas' [12] fisheye view approach.

Furnas introduced the concept of generalized fisheye views based on the observation that "humans often represent their own 'neighborhood' in great detail, yet only major landmarks further away. This suggests that such views ('fisheye views') might be useful for the computer display of large information structures like programs, data bases, online text, etc." in providing detail embedded in context. In the original paper by Furnas [12], the basic motivation for fisheye views was described, and the "degree of interest (DOI) functions" concept was introduced to formalize generalized fisheye views. According to his formulation:

$$\mathrm{DOI}_{\mathrm{fisheye}}(x,y) = \mathrm{API}(x) - D(x,y) \quad (5)$$

where $\mathrm{DOI}_{\mathrm{fisheye}}$ is the user's degree of interest in point, x, given that the current point of focus is y, API(x) is then given a priori importance of x, and $D(x,y)$ is the distance between x and the current point y.

Using this formulation, Furnas defined an application for tree structures and a specific example for tree structured text files.

As explained in Leung and Apperley [24], there are a number of ways a fisheye view can be created. Sarkar and Brown [34] used a formulation similar to the original one introduced by Furnas and applied the fisheye view technique for viewing and browsing computer graphs. They introduced layout concepts to formalize fisheye views, and built a framework to incorporate arbitrary structures by redefining the "distance" concept.

Lamping and Rao [22] describe an implementation for presenting a 2-D graph through a fisheye zoom. The hyperbolic browser provides a smoothly varying "focus + context" view; the display space allocated to a node decreases continuously with the distance from the focus, yet does not disappear abruptly. The Star Tree™ system from Inxight Software (http://www.inxight.com) is based on this principle and produces maps, which display "details + context" views of hierarchical structures including web sites.

2.4. Summary

As evident from the brief discussion in this section, use of clustering and visualization has found applicability in reducing overload, especially in the examination of complex information. Also, the idea to smoothly integrate context and details by means of distortion-oriented, or more specifically fisheye view, systems is promising. To our knowledge, use of clustering supported with fisheye views has not been applied to the visualization of web search results, hence the motivation for this study, which aims to build a web search aid that has a clustering-based visual presentation system with a fisheye capability.

Our contribution in this research thus is an original design in the visualization step, that is, f_4 or f_4', within our formal specification of the problem. Therefore, in Section 3, we describe our algorithm for the visualization component of our system. Then in Section 4, we describe a system that integrates the visualization

component with other components providing the rest of the functionality described above.

3. The "zoomable treemap" algorithm

The "zoomable treemap" algorithm is our modification of the treemap algorithm by Johnson and Shneiderman [17] and Shneiderman [37] with extended features to make the visualizations amenable to a fish-eye zoom. In this section, we describe "zoomable treemap," starting with the original treemap algorithm. The discussion in Section 3.1 provides an understanding of the basics of the technique. Then in Section 3.2, we describe the original features of our approach.

3.1. The original treemap algorithm

The treemap algorithm [17,37] stems from the observation that despite its intuitive appeal, presentation of a hierarchy as a tree is space-inefficient, and is therefore not appropriate for large hierarchies. As discussed in Section 2, maps are popular presentations of an information space. The treemap technique presents a hierarchy by means of a two-dimensional map. Consequently, it uniquely integrates a popular way of organizing information (hierarchical clustering) with a popular way of displaying it (two-dimensional maps).

The basic theme of the algorithm is a simple "slice-and-dice" idea. The available map space is first sliced vertically with every slice having an area proportional to the weight of the first-level cluster it represents. These weights can be defined differently according to the requirements of the specific application. For example, in the visualization of a computer file structure, these weights can be defined as the size of a file, or the total size of the files in a directory. In our application, the initial weight of a cluster is defined as the total number of documents in the cluster and this weight is adjusted while zooming. After the first-level slicing, a similar partitioning is applied to each vertical slice, this time creating horizontal slices of lower level clusters where again the areas of these slices are proportional to the weights of the corresponding subclusters.

For a simple illustration of the technique, see the hierarchy depicted in Fig. 2. Assuming that the lowest level nodes C, D, E, H, I, J, and K are all individual files with a weight of 1, the weights of the clusters will be 3 for A, 4 for B, and 2 for each of F and G. According to this weight structure, a treemap of the hierarchy is formed as follows: initially the map space is divided into two because there are only two first-level clusters: A and B. The ratio of the area allocated for A to the area allocated for B will be 3:4 because A has a weight of 3 and B has a weight of 4. After the first slicing, the areas for clusters A and B are partitioned similarly where the area of A is divided into three equal portions (between C, D, and E), and the area for B is divided into two equal portions (between F and G). Finally, the areas for F and G are both divided into two equal portions to represent H and I, and J and K, respectively. The resulting treemap is shown in Fig. 3.

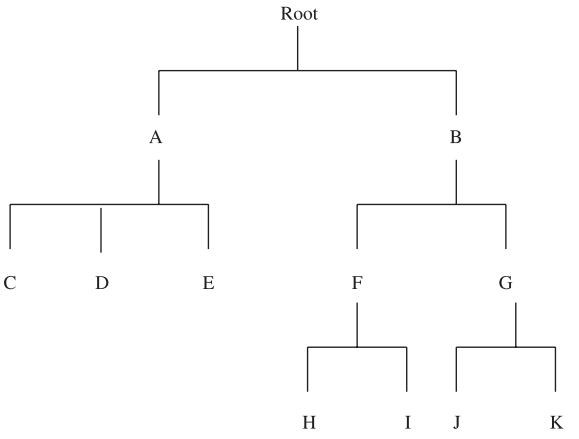

Fig. 2. A simple hierarchy.

3.2. The additional features of "zoomable treemap"

In the original treemap algorithm, the partitioning of the available map space continues until each individual document is represented on the map. Such a visualization can be very overwhelming for substantially large collections. For this reason, our algorithm stops the partitioning before the area of a slice drops below a predefined visibility threshold. For example, in the treemap displayed in Fig. 3, each region that is allocated to one of the lowest level nodes in the hierarchy, that is, C, D, E, H, I, J, and K, is one seventh, or approximately 14% of the total available map space. Let us assume that the predefined threshold for this map is 0.2, which means that a cluster will only be visible if its area is larger than 20% of the total map space. In this

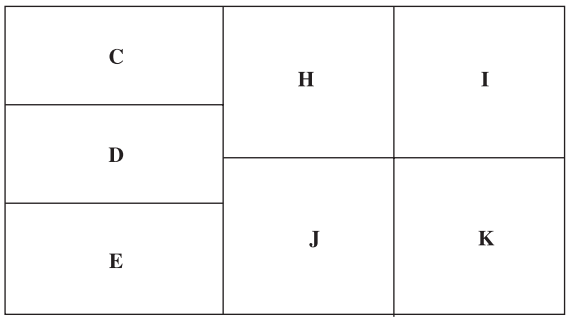

Fig. 3. The treemap of the hierarchy in Fig. 2.

case, all of these lowest level nodes would be below the threshold ($0.14 < 0.20$); therefore, the partitioning should stop before the lowest level. The result is shown in Fig. 4. The caveat to this approach is that there may be cases where some first-level clusters will always be too small, and will never be visible if this threshold is strictly enforced. Accordingly, our approach ignores the threshold for first-level clusters.

The original treemap algorithm shows every detail of the information structure in one visualization. Thus, there are no portions of the overview, details of which require further zooming. However, as seen in the map of Fig. 4, our approach is to make the clusters visible only if they are "large enough." For this reason, there is a need for zooming to visualize the details of a cluster not available on the overview. The design of full zooming (strict filtering) into our system would be straightforward, in that it simply entails redrawing the treemap using the zoomed-in node (cluster) as the root. This is equivalent to assigning a weight of zero to all out-of-focus clusters and redrawing the map. Going back to our example in Fig. 4, "full" zooming on the region labeled "A" on this map would result in the simple structure displayed in Fig. 5.

Our approach to fisheye zooming is one of many possible interpretations of the idea. We propose a method of zooming that requires the area of every cluster to be redefined by increasing the weight, or more specifically the degree of interest, of the zoomed-in cluster and all of its descendents such that all "children" of the zoomed-in cluster become visible, that is, exceed the visibility threshold. This means that the size of the smallest "child" of the zoomed-in cluster will determine the scaling factor. Given DOI is the degree of interest; we denote the degrees of interest for the overall collection, the zoomed-in node, and its smallest subcluster (child) as DOI_0, DOI_Z, and $DOI_{small\ child}$, respectively. Then for the smallest child to reach the threshold after the scaling, the following equation should be satisfied:

$$\text{threshold} = (DOI_{small\ child} \times \text{SCALE}) \Big/ ((DOI_Z \times \text{SCALE}) + DOI_0 - DOI_Z) \quad (6)$$

From Eq. (6), the scale factor can be calculated easily as follows:

$$\text{SCALE} = \text{threshold}\,(DOI_0 - DOI_Z) \Big/ (DOI_{small\ child} - (\text{threshold} \times DOI_{Pi})) \quad (7)$$

Note that the denominator of the right hand side of Eq. (7) will be zero for a scale that would effectively result in full zoom, and negative if the smallest "child" is too small to be visible even with the highest possible scale factor. In such cases, the threshold is ignored for

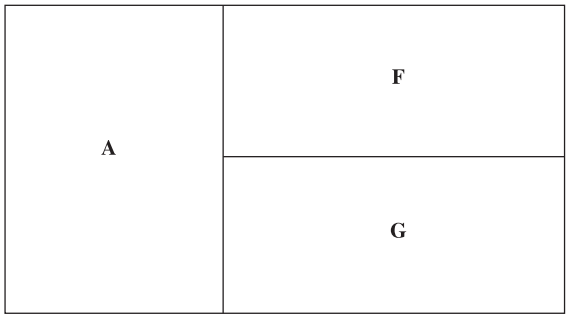

Fig. 4. Zoomable version of the treemap in Fig. 3.

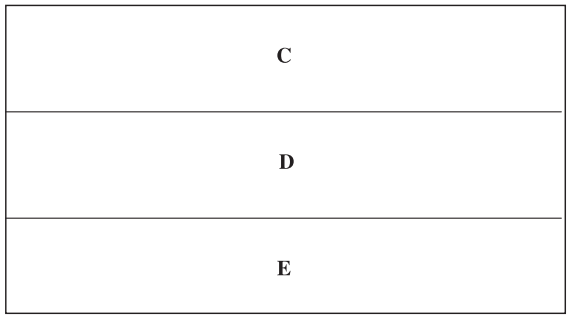

Fig. 5. "Full" zooming on region A in Fig. 4.

that cluster, and it is made visible. The scale factor is then calculated in a way to make the next to the smallest subcluster exceed the visibility threshold, that is, a positive "finite" scale factor, if possible. This process is repeated until a large enough subcluster is found, and the scale factor is calculated accordingly. Meanwhile, the out-of-zoom clusters are allocated less space because the ratio of their weights to the weight of the overall hierarchy decreases. By this decrease, the areas of some of the out-of-zoom clusters may drop below the visibility threshold. In case these clusters do not have a sibling in zoom, the algorithm modifies the view such that only the parents of the clusters below the visibility threshold are displayed. This way, the out-of-zoom area is summarized to provide a context within which the zoomed-in details are visible. On the other hand, if an out-of-zoom cluster is below the threshold and one of its siblings is in zoom, it cannot be combined with its other siblings, as that would not allow the zoomed-in sibling to be visible. In that case, the cluster below the threshold is simply displayed as it is.

Continuing on with our example, fisheye zooming on region A would require that the algorithm increase the degree of interest to a level that the areas of these regions would be 20% of the overall map space. From Eq. (7), it could be shown that a scale factor of 2 would increase the areas of these regions enough to make them exceed the visibility threshold. The resulting areas would be 20% of the map space for all C, D, E, F, and G. Note that the map space allocated to F and G would drop due this adjustment. Next, the algorithm checks whether this drop brings the regions below the visibility threshold. With the scale factor 2, the degrees of interest for C, D, and E increase to 2, and those for F and G remain at 2. Accordingly, F and G are still above the threshold although their relative areas shrank. The resulting map after these adjustments is located in Fig. 6.

After a cluster is zoomed as described, the viewer of the visualization may be interested in one of the subclusters thereof and want to further explore that subcluster, or may focus on an out-of-zoom cluster. In the former case, the weights are adjusted exactly as they were done in the previous zoom. In the latter case, the weights of the zoomed-in clusters from the previous zoom are reset (i.e. set to their originals) before the procedure of weight adjustment is performed for the new zoomed-in cluster. The reason for this extra step is the need to make the visualizations independent of the previously visited clusters unless the browsed area is still inside those clusters. Fig. 7 summarizes the visualization algorithms.

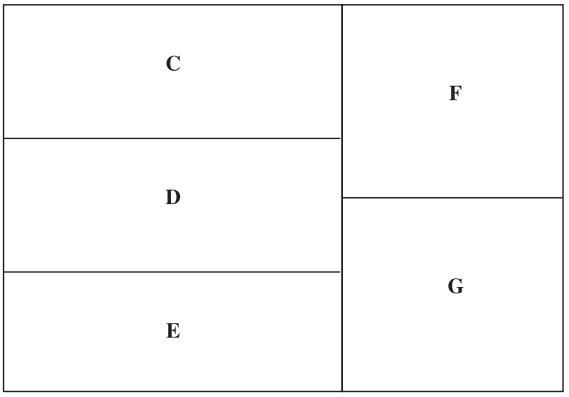

Fig. 6. "Fisheye" zooming on region A in Fig. 4.

4. Development of the fisheye-based information search processing aid (FISPA)

The input to our search processing aid is a list of web search results from AltaVista. AltaVista is a popular commercial search engine from Digital Corporation. Our system could work with other commercial search engines just as well. AltaVista's flexibility in accepting differing forms of queries, and the relative ease with which the list of search results can be parsed for further processing influenced our decision to choose it at the back end of our system. The design of our prototype represents a specific set of decisions in the selection of the technologies in the various steps of the model displayed in Fig. 1, and described in Section 2. Below is a brief description of the system components that perform these tasks.

4.1. System components

4.1.1. Document representation and clustering

IBM's "IntelligentMiner for Text" tool was used for this part of our system. IntelligentMiner is commercially available as a fast and easy-to-use tool. It uses lexical affinities [40] instead of single words in indexing a group of documents. This way, the vector size is more manageable, which improves the speed of the

algorithm. For our specific application, hierarchical clustering is preferred to nonhierarchical clustering for the following reasons:

- Nonhierarchical algorithms require some parameters to be specified in advance assuming that there is some prior knowledge about the vector space. In an automated system, this kind of prior knowledge is not always available.
- When the number of objects to be clustered is high, nonhierarchical clustering algorithms either create too many clusters, or they group too many objects in one cluster. This kind of a structure may not reduce the overload to a desired level.

Notation:

D_i : Degree of interest for node i
P_i : Parent node of i
$C_i^{\rightarrow}$: Vector of the children of node i
W_i : Weight of node i
DW_i : Drawing Weight of node i (W_i*D_i)
Len_i : Drawing Length of node i
Wid_i : Drawing Width of node i
$Left_i$: Position of the left side of the drawing for node i
Top_i : Position of the top side of the drawing for node i
$Level_i$: Level of node i in the hierarchy
($Level_0 = 1$)

Algorithm:

- For all i CALCULATE WEIGHT,
- RESET,
- PROCESS NODES,
- In case of user action:

ZOOM IN or RESETVIEW

CALCULATE WEIGHT routine
For all i recursively
If $C_i^{\rightarrow} = \varnothing$, then $W_i = 1$ (leaf node, weight is 1)
Else $W_i = \Sigma W_j$ where $j \in C_i^{\rightarrow}$ (sum of weights of children)

RESET routine
For all i
$D_i = 1$ (Drawing weights are equal to actual weights)

PROCESS NODES routine
- For all i $DW_i = W_i * D_i$
- For all i CALCULATE DRAWING DIMENSIONS
- For all i DECIDE NODE VISIBILITY
- For all i
If $visible_i$ = true then DRAW i

CALCULATE DRAWING DIMENSIONS routine
Len_0 = Length of the applet
Wid_0 = Width of the applet
$Left_0$ = Position of the left side of the applet
Top_0 = Position of the top side of the applet
(Position and dimensions of node 0 (root) are decided by the visible area of the applet.)
- For all i recursively;
If $Level_i$ is even

Fig. 7. The visualization algorithm with Fisheye zoom.

then $Top_i = Top_{Pi}$
$Len_i = Len_{Pi}$
$Wid_i = Wid_{Pi} * DWi / DW_{Pi}$
$Left_i = Left_{Pi} + \Sigma (Wid_j * done_j)$ where $j \in C_{Pi}^{\rightarrow}$.
(slice the parent vertically based on the proportional W_j's of the children).
Else
$Left_i = Left_{Pi}$
$Wid_i = Wid_{Pi}$
$Len_i = Len_{Pi} * DWi / DW_{Pi}$
$Top_i = Top_{Pi} + \Sigma(Len_j * done_j)$ where $j \in C_{Pi}^{\rightarrow}$.
(slice the node horizontally based on the proportional W_j's of the children).

DECIDE NODE VISIBILITY routine
- For all $i \in C_0^{\rightarrow}$ $visible_i$ = true
(The children of Node 0 (Root) are visible irrespective of their Wi's or *threshold*)
- For all $i \notin C_0^{\rightarrow}$ recursively;
If $(Len_i * Wid_i) / (Len_0 * Wid_0) >$ *threshold* then $visible_i$ = true
- For all $i \notin C_0^{\rightarrow}$ recursively;
If there exists $j \in C_{Pi}^{\rightarrow}$ for which $visible_j$ = false then $visible_i$ = false
(combine too small nodes with their siblings)
For all i recursively;If $visible_j$ = true for all $j \in C_i^{\rightarrow}$ then $visible_i$ = false
(If all the children of a node are visible, then the node becomes invisible)

ZOOM IN routine
- Find i that is selected
- If $C_i^{\rightarrow} = \varnothing$ (Empty vector, the node is leaf) then display the URL
Else
If $D_i = 1$ (node not in zoom) then RESET
(reset the drawing weights first if the zoom is switching)
SCALE = *threshold* * $(W_0 - W_{Pi}) / (W_{lightest\ child} - (threshold * W_{Pi}))$
$D_i = D_i * SCALE$
for all $j \in C_i^{\rightarrow}$ recursively:
$D_j = D_j * SCALE$
(scale up the size of all descendents)
PROCESS NODES

RESETVIEW routine
- RESET
- PROCESS NODES

Fig. 7 (*continued*).

- In a fisheye zoom system, as one looks at the information objects (pages) in one cluster, the other clusters need to be summarized to provide a context. This requires combining similar clusters at a higher level and hence assumes a hierarchy.

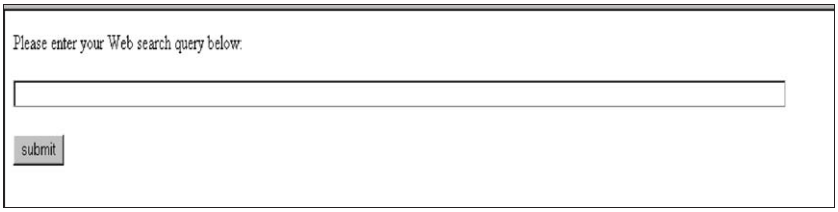

Fig. 8. The input form.

IntelligentMiner's hierarchical clustering component is based on the agglomerative strategy that is, considering each object as a cluster of its own and then iteratively joining clusters to form larger ones until there is only one cluster.

4.1.2. Visualization

This part of the system is of major interest to us for which we created our own component. This is a Java applet that visually displays a hierarchical structure based on the "zoomable treemap" algorithm described

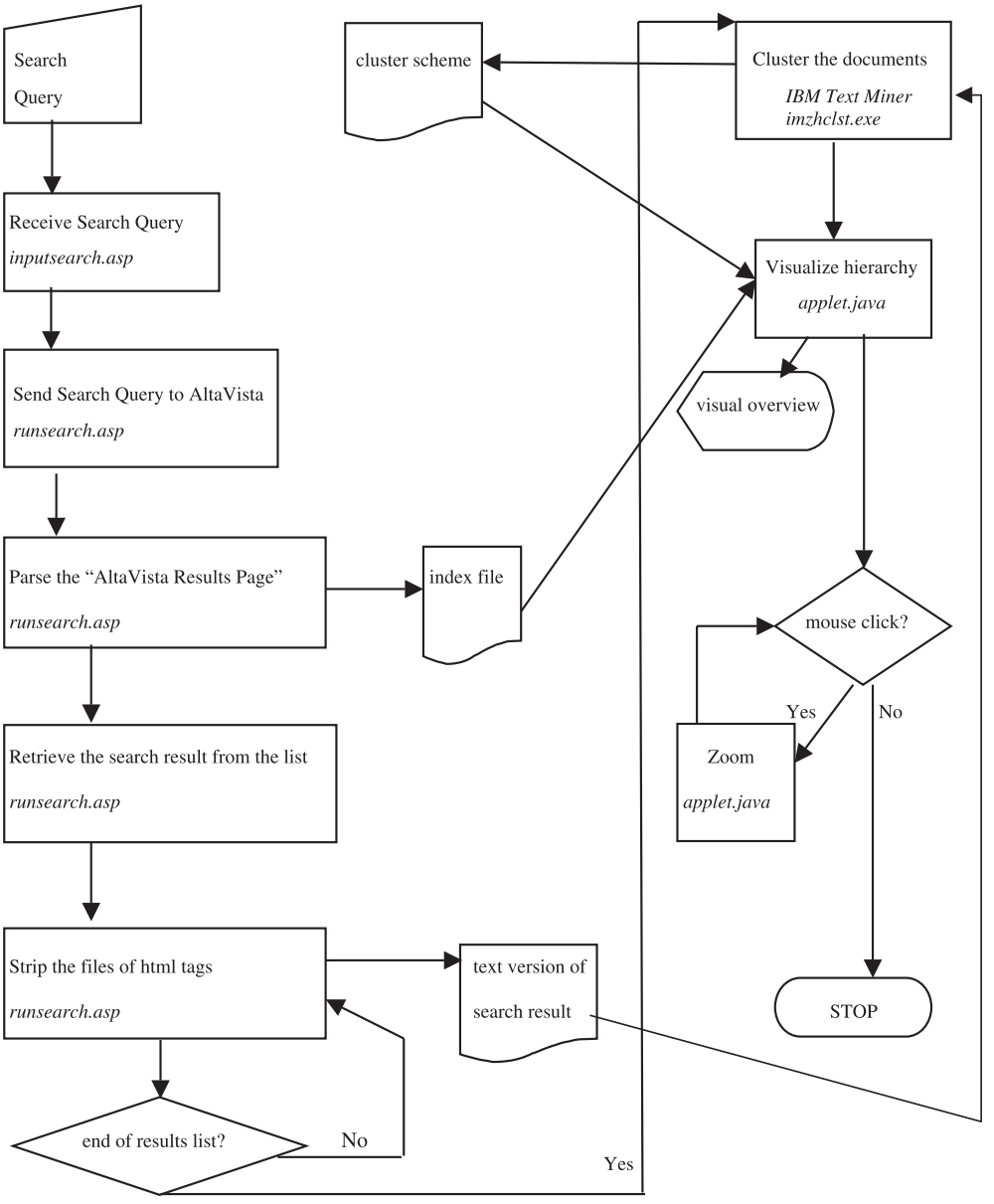

Fig. 9. The system flowchart.

in Section 3. This applet can visualize any hierarchy by first converting it into a data structure as an array of array objects, and then presenting an overview of this structure as a partitioned map as explained in Section 3. The overview map is "clickable" and responds to mouse clicks by invoking the zooming methods that were described in Section 3. Zooming is basically provided by displaying the "children," that is, sub-clusters or individual objects, which are one level below a cluster that it is clicked. When the area that corresponds to a leaf node is clicked, the prespecified attributes of the object are displayed. The functionality briefly described here applies to the visualization of a general hierarchy. In the next section, we discuss how we used this component specifically for visualizing web search results.

4.2. Integration and implementation with server-side scripting

We implemented a working prototype system as a proof of concept for the design ideas discussed in the previous parts of this section. The system facilitates the processes displayed in Fig. 1, and performs them sequentially. A simple active server pages (ASP) form (see Fig. 8) was designed to let the web searcher enter a query. This form sends the query to an ASP document that communicates with the AltaVista search engine. The query is sent to AltaVista in a format that will allow us to receive the first 100 search results. Next, an ASP script parses the "AltaVista Results" using a component (ASPTear), which has the built-in functionality to parse, or "tear" a given HTML document. This way, the title and URL of each search result is extracted from the "AltaVista Results" page. The name (given by the script for internal use by the system's clustering component), the title, and the URL of each of the 100 search results are stored in an index file. Next, another ASP script retrieves each of these results by calling their URLs. The resulting documents are in html format; therefore, they are first converted to text files by stripping them of their html tags. The resulting text files are locally saved under their unique names already in the index file. The clustering component (Intelli-

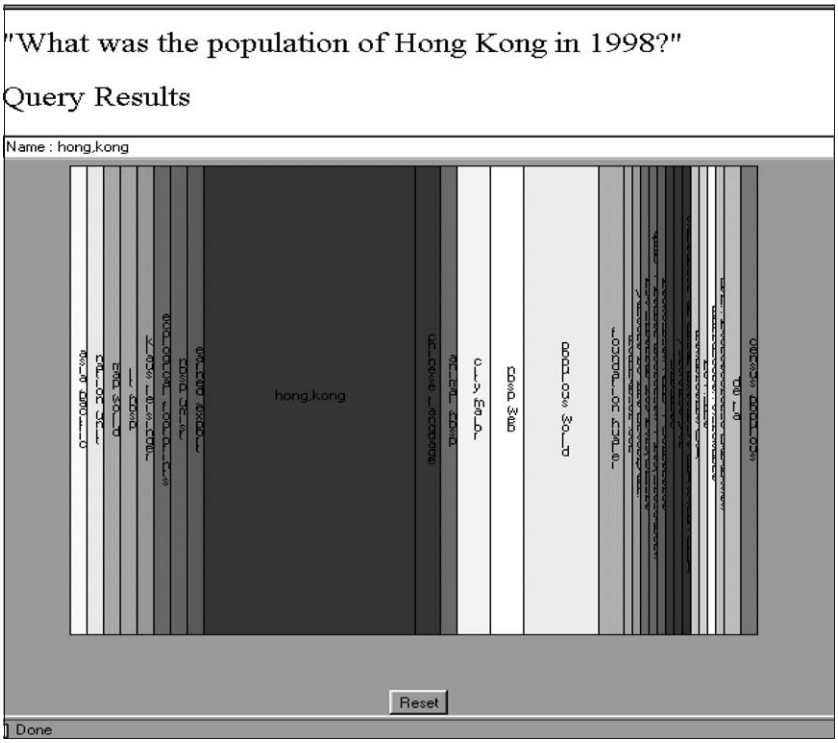

Fig. 10. Visual overview of the search results to the query "What was the population of Hong Kong in 1998?".

gentMiner's hierarchical clustering routine) works only with these names for creating a hierarchical cluster scheme. This cluster scheme is saved to a file, and is sent to the visualization component.

Next, the visualization component, that is, the Java applet (we refer to this applet as applet.java in the following discussions), reads the index file and the hierarchical cluster information from the output of the clus- tering routine. In displaying the hierarchy, applet.java uses the output of the clustering routine to label the clusters. Leaves of the hierarchy, that is, individual files, are labeled by looking up the corresponding title for each file name in the index file that was created before.

As mentioned before, the user can click on a part of the overview for zooming. When a leaf, that is, an individual file, is reached, the applet looks up the URL of the file from the index file, retrieves the page, and displays it in a separate window. Unless an individual page is retrieved as described, the only difference between the visual presentations of clusters and individual pages is that individual pages occupy less space than a cluster at the same level of the hierarchy because they have minimal weights (1).

Fig. 9 summarizes the technical schematic of our system. This summary includes the main processes, the inputs and outputs produced, and the specific program segments that execute each of the processes. Two of these processes, namely, "Visualize Hierarchy" and "Zoom," are performed by means of applet.java, the implementation of which is based on the algorithm in Fig. 7.

5. A sample session

As described previously, the users of our system enter their queries through the form displayed in Fig.

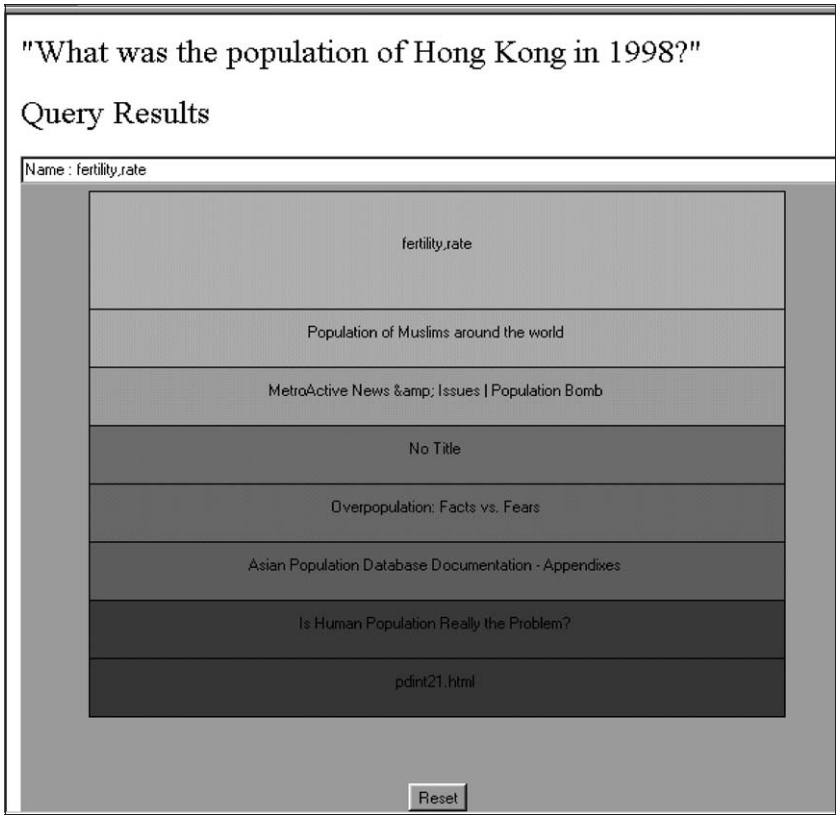

Fig. 11. "Full" zooming of the "populous world" section of Fig. 10.

8. Fig. 10 displays the visual overview that our system provides for the example query, "What was the population of Hong Kong in 1998?" As seen in Fig. 10, the largest group of the results to this query is labeled "Hong Kong," where some other groups are "populous world" and "city map."

Let us assume that the user thinks the answer they are seeking is under the "populous world" section of this map. Then the next step would be to zoom on that portion of the map. Fig. 11 displays the resulting screen if the user is interacting with the "full-zoom" version of our system, which is based on the traditional full-zoom idea; details are displayed without any context information.

Fig. 12 displays the alternative "fisheye view" of the same portion of the map. The details of this cluster are now being displayed without totally eliminating the rest of the overall map. Note that some of the groups on this view are very small and, therefore, are difficult to recognize. This problem occurs when the clustering algorithm creates too many first-level clusters that cannot be combined with others to provide a more summarized representation of the context. While this is a problem with the clustering rather than the visualization algorithm, the user should still be supported with clues as to what each of these regions on this map are. The way we chose to provide such support is by means of implementing a mouse-over function where the users can see the name of the cluster their mouse is over in the title bar right above the map. Also, note that this problem is not specific to the fisheye implementation, because the small clusters are hard to recognize even at the first-level overview.

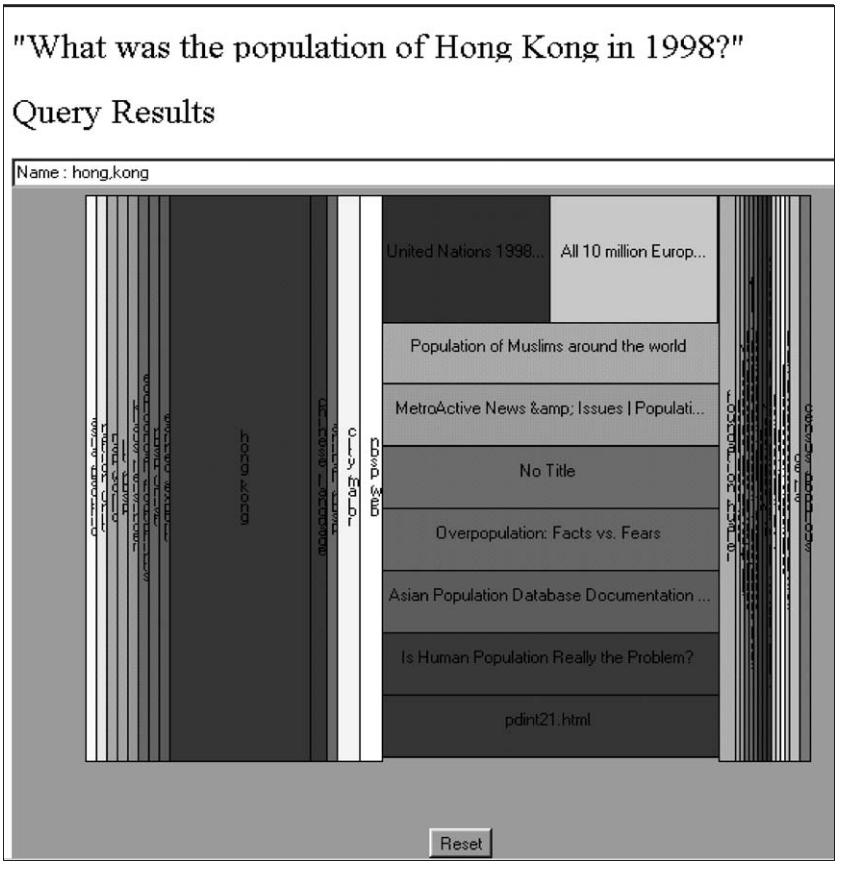

Fig. 12. "Fisheye" zooming of the "populous world" section of Fig. 10.

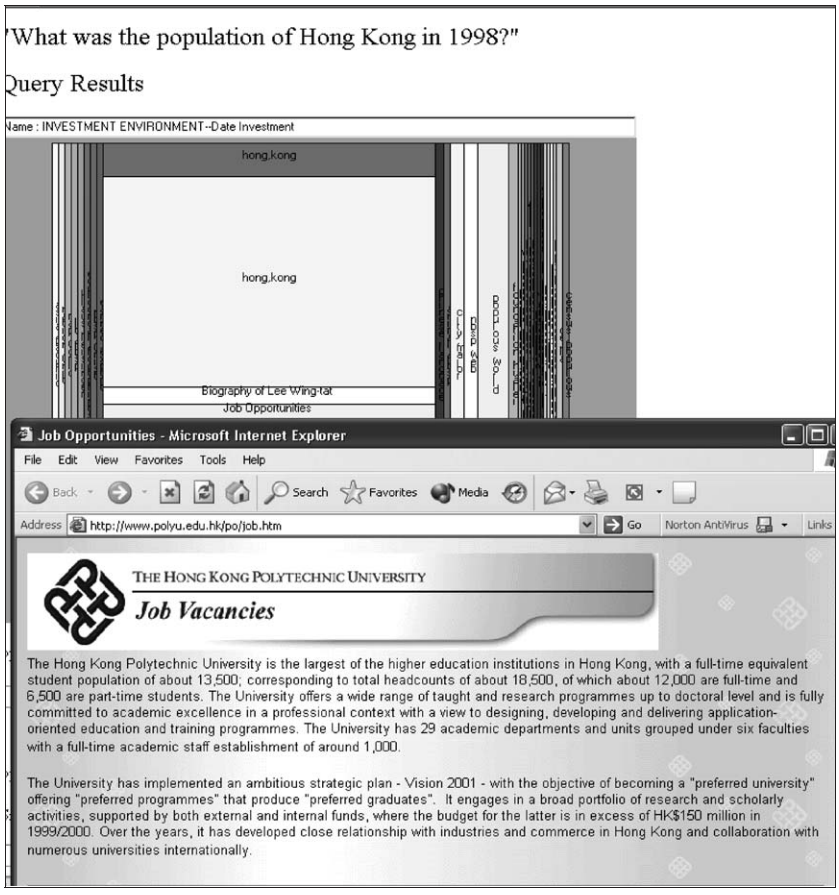

Fig. 13. Looking into the contents of the web page titled "Job Opportunities".

The user interaction with the interface continues in this manner where the user moves back and forth between different clusters and web pages until they find the answer they are seeking. Otherwise, the process is repeated with a new search query. Fig. 13 displays the screen that the user will see when they are looking into the contents of an individual page titled "Job Opportunities" in its separate window.

6. Usability studies

We conducted several sessions of experiments to test the usability of our systems. The sessions resulted in a total sample of 57 subjects who were undergraduate and graduate level business (mostly MIS) students. The goal of the experiments was to compare the visual systems (FISPA and its full-zoom counterpart) to the text-based system, and to compare the full-zoom visualization to the fisheye zoom visualization in terms of effectiveness (number of correctly answered questions) and efficiency (speed in answering the questions). We asked the subjects three groups of questions with four general-interest questions in each group. The users were supported by a different mode of presentation, that is, text-based, full zoom, or fisheye zoom for each of the question groups. Therefore, all subjects answered all questions, and used all three systems, but had different combinations of system and question-group. The time it took for the subjects to finish each section was automatically

recorded. After the data were collected, we analyzed the records and assigned a score to each subject for every experimental phase depending on how many correct answers they found for the questions in that phase.

The data analyses (MANOVA) showed that the subjects were significantly faster with the visual systems than they were with the text-based system ($p = 0.00$) while their scores were not significantly different ($p = 0.48$). Similarly, the fisheye system resulted in significantly faster performance than the full-zoom system ($p = 0.04$) without any significant change in scores ($p = 0.12$). Full details of the usability studies and the analysis and implications of the results are described in Ref. [42].

In addition, the comments we received from some of the experimental subjects such as "Excellent job! It is time-saving and easy" were encouraging as to the user-acceptance of our design ideas. Meanwhile, our observations during the usability studies and our own experiences with the use of the prototype suggest how the system can be improved. We discuss these issues in the next section.

7. Discussion

A growing conviction exists in the system design community that user interface is crucial. Interface development will gain more importance given that due to the increasing number of novice information technology users, the trend in the computing world is toward more user-friendly systems. From that perspective, we believe that our effort in the development of a system with a novel interface that addresses a real problem in the popular activity of web search is valuable. Visual aids such as windows, frames, icons, and images are already inevitable elements of user-friendly interface design. Thus, the general idea of using visualization in interface design has great potential, especially with the growing image processing capabilities of today's computers.

The building of the prototype system involved many decisions on the selection or implementation of the system components. These decisions were determinants of the quality of the prototype system, and are worth discussing for identifying possible improvement efforts.

Although not in depth, our brief evaluation of web search engines led us to the conclusion that the Northern Light search engine is interesting in that it provides an alternative clustering-based presentation of search results. Considering our approach is based on the clustering of search results, the clusters that are readily available from Northern Light can be used for visualization purposes. Incorporation of a search engine that includes clustering will add the speed the system needs to find a more practical applicability. On the other hand, the idea of presenting clusters of search results can easily be adopted by other commercial search engines such as the recently popular Goggle. A search engine that chooses to include the clustering feature as part of its system can adopt our visual design ideas, and implement a system with better integration of the individual components. Such an integrative approach would yield higher system efficiency.

Another point that is related to the speed of the system is the approach that was followed to represent each search result. Currently, the system follows the hyperlink from each result to retrieve the whole web page, and saves the text portion of those pages locally as text documents. Because it is prone to network traffic and the performance of the communicating servers, the retrieval process is the most time-consuming component of the system. Our reason for following this approach was the conviction that the quality of the resulting clusters would be higher if we used whole pages in clustering instead of using only the title or a summary for each result. We are aware of the speed degradation such an approach would lead to, yet for the time being, our focus was on the visualization component, and not the speed of the overall system.

Zamir and Etzioni [45] found that using snippets instead of full documents for clustering did not lead to significantly lower cluster quality while improving the clustering speed considerably. Accordingly, a speed improvement in our system can be achieved by using the short snippet provided by the search engine in the "results" page, instead of following the links and retrieving the whole documents. Experimenting with different forms of input to the clustering process is an interesting activity, which needs to be further studied.

Our system uses a tool that utilizes one of the numerous possible techniques for clustering docu-

ments. We observed some problems with the quality of the clusters that this tool created. Our choice of IntelligentMiner was mainly based on convenience, that is, availability and accessibility. IntelligentMiner is very fast in creating a cluster structure and needs no further processing in the way it presents its output. Nevertheless, we believe that most of the shortcomings of the system were due to the fact that the clustering structures created were not as useful as expected. Although part of the reason for this is that the input to the clustering algorithm, that is, the search results, may at times be totally unrelated, we still believe that some of these problems can be remedied by experimenting with different clustering techniques. This is an open issue, which requires further investigation, and we shall pay specific attention to this particular point in our future research.

Our research reported in this paper involves experimentation with a specific, original implementation of the fisheye view technique. Consequently, the insights gained from the development effort have only limited generalizability as to the success of the fisheye zooming, or in general terms, the "details embedded in context" idea. It is our conviction that the general principles of fisheye views are fairly applicable in the specific domain of application we addressed, and further research should explore the application of different techniques for fisheye zooming. A natural extension of this idea would be the exploration of different methods to create visual overviews of the document hierarchy, and compare them to the map representation implemented in this study. An example method of this kind is the hyperbolic tree we discussed in Section 2. Hyperbolic trees are particularly good fisheye view implementations, although their applicability to very large collections is debatable because the visual presentations produced using hyperbolic trees are not space efficient. However, it would still be interesting to compare this method to the one we developed and reported in this study.

Meanwhile, it should be noted that system quality is not the only determinant of the usability of a system. User characteristics and behavioral variables such as commitment and reaction to change should also be considered in explaining the successful use of a system. Therefore, we believe that the usability of our system can be realistically evaluated only after a period of continuous use where the users are familiar with it and the initial performance problems are solved. Nevertheless, the results of our usability studies show that there is value in using the system, and it is worth pursuing the suggested enhancements to make it more widely accepted by potential users.

Acknowledgements

We appreciate the cooperation and access to Treemap97 code from Ben Shneiderman and the Human–Computer Interaction Lab at the University of Maryland. We are very thankful to Mr. Someshwar Baldawa for his contribution in the coding of the algorithm. Also, Mr. Vishal Jangla and Mr. Sanjay Vanketasvarulu are highly appreciated for the design and coding of the server-side scripts.

References

[1] R.A. Botafogo, E. Rivlin, B. Shneiderman, Structural analysis of hypertexts: identifying hierarchies and useful metrics, ACM Transactions on Information Systems 10 (2) (1992) 142–180.

[2] S.K. Card, G.G. Robertson, W. York, The WebBook and the Web Forager: an information workspace for the World Wide Web, Proceedings of the Conference on Human Factors and Computing Systems, April 13–18, 1996, Vancouver, British Columbia, 1996.

[3] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, J. Kleinberg, Automatic resource compilation of analyzing hyperlink structure and associated text, http://decweb.ethz.ch/WWW7/1898/com1898.htm downloaded (April 24th, 2002), 1996.

[4] H. Chen, A.L. Houston, R.R. Sewell, B.R. Shatz, Internet browsing and searching: user evaluations of category map and concept space techniques, Journal of the American Society for Information Science 49 (7) (1997) 582–603.

[5] E.C. Chewning Jr., A.M. Harrell, The effect of information load on decision makers' cue utilization levels and decision quality in a financial distress decision task, Accounting, Organizations and Society 15 (6) (1990) 527–542.

[6] D.R. Cutting, D.R. Karger, J.O. Pedersen, J.W. Tukey, Scatter/Gather: a cluster-based approach to browsing large document collections, Proceedings of the 1992 Conference on Research and Development in Information Retrieval, June 21–24, 1992, Copenhagen, Denmark, 1992.

[7] D.R. Cutting, D.R. Karger, J.O. Pedersen, Constant interaction-time Scatter/Gather browsing of very large document collections, Proceedings of the Fifteenth Annual International

ACM Conference on Research and Development in Information Retrieval, June 27–30, 1993, Pittsburgh, PA, 1993.
[8] P. De Bra, G.J. Houben, F. Dignum, Task-based information filtering: providing information that is right for the job, http://wwwis.win.tue.nl/~houben/respub/infwet97.html downloaded (April 23rd, 2002), 1997.
[9] A. Dieberger, A.U. Frank, A city metaphor for supporting navigation in complex information spaces, Journal of Visual Languages and Computing (9) (1998) 597–622.
[10] K.L. Fox, O. Frieder, M.M. Knepper, E.J. Snowberg, SENTINEL: a multiple engine information retrieval and visualization system, Journal of the American Society for Information Science 50 (7) (1999) 616–625.
[11] J.C. French, D.E. Brown, N.H. Kim, A classification approach to Boolean query reformulation, Journal of the American Society for Information Science 48 (8) (1997) 694–706.
[12] G.W. Furnas, Generalized fisheye views, Proceedings of the Conference on Human Factors in Computing Systems, April 16–23, 1986, Boston, MA, 1986.
[13] G. Golovchinsky, M.H. Chignell, The newspaper as an information exploration metaphor, Journal of Information Processing and Management 33 (5) (1997) 663–683.
[14] M. Hearst, TileBars: visualization of term distribution information in full text information access, Proceedings of the Conference on Human Factors in Computing Systems, May 7–11, 1995, Denver, CO, 1995.
[15] M. Hearst, P. Pedersen, Reexamining the cluster hypothesis: scatter/gather on retrieval results, Proceedings of the Conference on Research and Development in Information Retrieval, August 18–22, 1996, Zurich, Switzerland, 1996.
[16] I. Herman, G. Melancon, M.S. Marshall, Graph visualization and navigation in information visualization: a survey, IEEE Transactions on Visualization and Computer Graphics 6 (1) (2000) 24–43.
[17] B. Johnson, B. Shneiderman, Treemaps: a space filling approach to the visualization of hierarchical information structures, Proceedings of the Second International IEEE Visualization Conference.
[18] K.L. Keller, R. Staelin, Effects of quality and quantity of information on decision effectiveness, Journal of Consumer Research 14 (2) (1987) 200–213.
[19] T. Kohonen, Self-organizing maps, Springer Series in Information Sciences, New York, NY, 2001.
[20] H.P. Kumar, C. Plaisant, B. Shneiderman, Browsing hierarchical data with multi-level dynamic queries and pruning, International Journal of Human-Computer Studies 46 (1) (1997) 103–124.
[21] K. Lagus, S. Kaski, T. Honkela, T. Kohonen, Browsing digital libraries with the aid of self-organizing maps, Proceedings of the Fifth International World Wide Web Conference, May 6–10, 1996, Paris, France, 1996.
[22] J. Lamping, R. Rao, Laying out and visualizing large trees using a hyperbolic space, Proceedings of the ACM Symposium on User Interface Software and Technology, November 2–4, 1994, Marina del Rey, CA, USA, 1994.
[23] J. Lamping, R. Rao, Visualizing large trees using the hyperbolic browser, Proceedings of the Conference Companion on Human Factors and Computing Systems, April 14–18, 1996, Vancouver, British Columbia, 1996.
[24] K.Y. Leung, M.D. Apperley, A review and taxonomy of distortion-oriented presentation techniques, ACM Transactions on Computer-Human Interaction 1 (2) (1994) 126–160.
[25] X. Lin, Map displays of information retrieval, Journal of the American Society for Information Science 48 (1) (1997) 40–54.
[26] N.K. Malhotra, Information load and consumer decision making, Journal of Consumer Research 8 (4) (1982) 419–430.
[27] D.A. Nation, C. Plaisant, G. Marchionni, A. Komlodi, Visualizing web sites using a hierarchical table of contents browser: WebTOC, Proceedings of the Third Conference on Human Factors and the Web, April 1998, Los Angeles, CA, 1998.
[28] P. Pirolli, P. Schank, M.A. Hearst, C. Diehl, Scatter/Gather browsing communicates the topics structure of a very large text collection, Proceedings of the ACM Conference on Human Factors in Computing Systems, April 13–18, Vancouver, British Columbia, Canada.
[29] E. Rivlin, R.A. Botafogo, B. Shneiderman, Navigating in hyperspace: designing a structure-based toolbox, Communications of the ACM 37 (2) (1994) 87–96.
[30] G.G. Robertson, J.D. Mackinlay, S.K. Card, Cone trees: animated 3D visualizations of hierarchical information, Proceedings of the ACM Conference on Human Factors in Computing Systems.
[31] D. Roussinov, H. Chen, Information navigation on the web by clustering and summarizing query results, Information Processing & Management 37 (6) (2001) 789–816.
[32] I.A. Rudy, A critical review of research on electronic mail, European Journal of Information Systems 4 (4) (1996) 198–213.
[33] G. Salton, Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer, Addison-Wesley, Reading, MA, 1989.
[34] M. Sarkar, M.H. Brown, Graphical fisheye views of graphs, Proceedings of the Conference on Human Factors in Computing Systems, May 3–7, 1992, Monterey, CA, USA, 1992.
[35] J. Savoy, Ranking schemes in hybrid boolean systems: a new approach, Journal of the American Society for Information Science 48 (3) (1997) 235–253.
[36] D. Setton, Information overload, Forbes 160 (6) (1997) 18–20.
[37] B. Shneiderman, Tree visualization with treemaps: a 2D space-filing approach, ACM Transactions on Graphics 11 (1) (1992) 92–99.
[38] B. Shneiderman, The eyes have it: a task by data type taxonomy of information visualizations, Proceedings of the IEEE Symposium on Visual Languages, September 03–06, 1996, Boulder, CO, 1996.
[39] B. Shneiderman, Designing The User Interface, Strategies for Effective Human–Computer Interaction, Addison Wesley, Reading, MA, 1997.
[40] D. Tkach, Technology text mining—turning information into knowledge—a white paper from IBM, http://www-3.ibm.com/software/data/iminer/fortext/download/whiteweb.pdf downloaded (April, 24th, 2002), 1998.

[41] E.R. Tufte, The Visual Display of Quantitative Information, Graphics Press, Cheshire, CT, 2001.
[42] O. Turetken, R. Sharda, Visualization support for managing information overload in the web environment, Proceedings of the 22nd International Conference on Information Systems (ICIS), December 16–19, 2001, New Orleans, LA, USA, 2001.
[43] P. Willet, Recent trends in hierarchical document clustering: a critical review, Information Processing & Management 24 (5) (1988) 577–597.
[44] J.A. Wise, J.J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, V. Crow, Visualizing the non-visual: spatial analysis and interaction with information from text documents, Proceedings of the IEEE Information Visualization Symposium, October 30–31, 1995, Atlanta, GA, USA, 1995.
[45] O. Zamir, O. Etzioni, Grouper: a dynamic clustering interface to web search results, Proceedings of the 8th International World Wide Web Conference, May 11–14, 1999, Toronto, Canada, 1999.

Ozgur Turetken is Assistant Professor of Management Information Systems in the Fox School of Business and Management at Temple University. He received a BS in Electrical Engineering and an MBA from Middle East Technical University in Ankara, Turkey, and a PhD from Oklahoma State University. Dr. Turetken's research interests are in information visualization, decision support systems, and distributed work arrangements.

Ramesh Sharda is Conoco Chair of Management of Technology and a Regents Professor of Management Science and Information Systems in the College of Business Administration at Oklahoma State University. He received his BS Eng degree from University of Udaipur, MS from The Ohio State University, and an MBA and PhD from the University of Wisconsin-Madison. His research has been published in major journals in management science and information systems including *Management Science*, *Information Systems Research*, *Decision Support Systems*, *Interfaces*, *INFORMS Journal on Computing*, *Computers and Operations Research*, and many others. He served as the Founding Editor of the *Interactive Transactions of ORMS* and on the editorial boards of other journals such as the *INFORMS Journal on Computing*, *Information Systems Frontiers*, *Journal of End User Computing*, and *OR/MS Today.* One of his major activities in the last few years was to start the MS in Telecommunications Management Program at Oklahoma State. Now he is establishing a major interdisciplinary Institute for Research in Information Systems (IRIS) at OSU. His research interests are in optimization applications on desktop computers, information systems support collaborative applications, neural networks, business uses of the Internet, and knowledge networks. Defense Logistics Agency, NSF, Marketing Science Institute, and other organizations have funded his research. Ramesh is also a cofounder of a company that produces virtual trade fairs, iTradeFair.com.