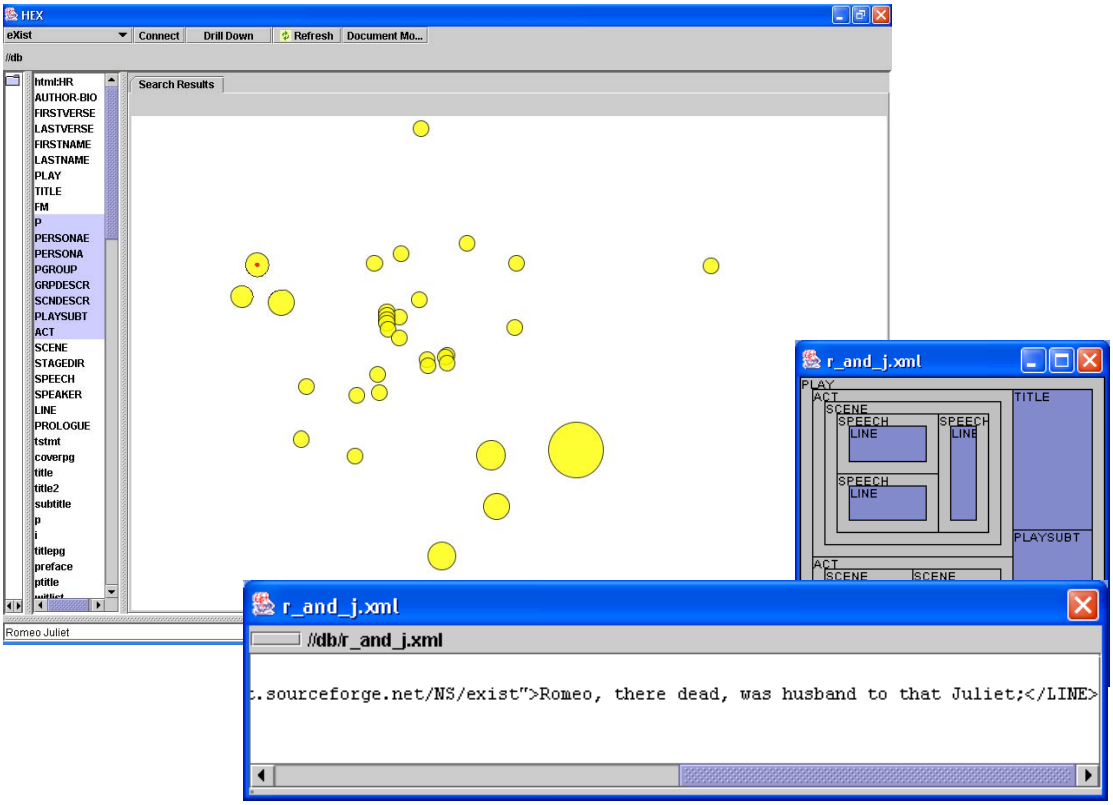
HEX: Visualizing **HE**terogeneous **X**ML Document Information Retrieval

534B Project
Matt Williams
46629960



1.0 Introduction

Searching for information on the Web's immense document collection lends itself to applying information visualization techniques as the document relevancy decisions can greatly benefit from involving the user in the loop. Visualization has been integrated in a variety of standard HTML Web documents but has yet to be found in systems searching through structured documents such as XML documents. The XML document structure provides both additional data organization and data description to standard Web documents and as XML becomes more pervasive, the need to take advantage of the XML's additional structure grows. However, the influx of the XML format also adds new challenges to Web IR systems over and above the problems of searching though massive collections of diverse content. The XML format provides great structural freedom and will produce great heterogeneity in document structure across different collections on the Web. XML IR systems that search across collections from multiple systems must account for diverse document structures. The goal of the HEX system presented in this paper is to improve information retrieval efficiency in large collections of diversely structured documents by applying multiple information visualization solutions.

Section 1 introduces the challenges presented by XML-IR that motivate the present system. Section 2 describes related research that has influenced the project. Section 3 describes the framework of the HEX system including the data model, the user interface, and the search operations. The paper finishes with an evaluation of the system, conclusion and a description of interesting future endeavors related to the HEX project.

1.1 Motivation

XML, short for eXtensible Markup Language, is extensible in the sense that it is not limited to a predefined language. Instead, XML allows any language for describing the data that is contained within. This offers the XML author great freedom in creating and organizing the data to fit individual needs. This is a great benefit for designers of specific systems that interchange data for a specific purpose (e.g. interchanging data on music collection catalogs) but it becomes a problem when trying to integrate or search for data across a number of separately designed systems. The freedom that XML offers can produce many various structures that all represent similar information. In many cases the designer must make arbitrary decisions about the number of elements, the particular element names, and the hierarchical structure of the elements. Figure 1 illustrates how two different structures can describe collections of musical CDs.

```
<ARTIST>
    <NAME> Beatles
    </NAME>
    <CDCOLLECTION>
    <CD>
        <TITLE> Yellow Submarine
        </TITLE>
        <SONG> Yellow Submarine
        </SONG>
        <SONG>Help
        </SONG>
      ...
    </CD>
    ...
  </CDCOLLECTION>
<ARTIST>
```

```
<CDCOLLECTION>
    <CD>
        <ARTISTNAME> Beatles
        <ARTISTNAME>
        <TITLE> Yellow Submarine
        </TITLE>
        <SONG> Yellow Submarine
        </SONG>
        <SONG>Help
        </SONG>
      ...
    </CD>
    ...
</CDCOLLECTION>
```

Figure 1.

As the number of XML documents on the Internet grows, the interest in searching for data over large collections of heterogeneous XML data will mount. Moreover, due to the arbitrary structure design described above, the heterogeneity of the structures of XML documents with also increase dramatically.

The difficulty of information retrieval in such diverse data structures can be tackled in two ways. There exists a movement to reorganize data across systems into fewer structures, which is intended to reduce the challenge of diverse data structures (IEEE, 2001). However, the reorganization of such an extremely large data collection (i.e. the Web) may either be impossible or a very lengthy process. The alternative tries to cope with the heterogeneity by developing information retrieval systems with this diversity in mind that involve the user to help locate relevant documents. It is this second approach that motivates the HEX system presented in this paper. HEX improves information retrieval efficiency by organizing the results by similarity and applying various interactive visualization techniques to help the user navigate the results.

1.2 Contributions

Visualizing XML Document Search

To tackle the difficulty of navigation through an immense, highly diverse data set, the HEX system applies multiple information visualization techniques, including pan and zoom navigation, hierarchical structure visualizations, and large dataset visualization.

Overcoming Heterogeneity

Although a some information retrieval systems have been developed with the intention of handling diverse document structure (e.g. Theobald and Weikum, 2000; Fuhr and Grossjohann, 2000), none handle extremely divergent structures that arise from a simple arbitrary choice such as /AUTHOR/BOOK/PUBLISHER vs. /PUBLISHER/AUTHOR/BOOK. In these other systems, such structure differences would require the user have some a priori knowledge of the various structures to search effectively.

Iterative Search and Similarity

HEX attempts to handle such heterogeneity by treating the element names as a flat structure. Such a model may relax queries so much to the extent that unwanted data from a variety of domains may be returned. HEX attempts handle this difficulty in a number of ways such as clustering by similarity and iterative drill down methods. These methods will be described further in the later sections.

2.0 Related Work

2.1 Information Retrieval Visualization

As in the present system, there are numerous existing systems that group information retrieval results by similarity. The systems that categorize the results into folders (hearst and Pedersen, 1996; NorthernLight) are forced to specify the number of categories and the thresholds between the categories. To overcome the difficulties of categorization some systems provide a visualization of the results in 2 or 3 dimensional, thus allowing the user to visually compute the clusters of similarity. In 2001, Roussinov and Chen used Kohonen's self-organizing maps to organize and display Web search results.

Alternatively, Leuski and Allan (2000) developed the Lighthouse system that applied multi-dimensional scaling (MDS) to compute 2 or 3-dimensional similarity of web search results. The galaxies (Wise et al, 1994) system, in which documents are represented as stars in the galaxy of the document collection, was the first to visualize document similarity using MDS. To my knowledge, no system applies such techniques in XML or other structured document domains.

The present system adds to the galaxy representation by including abstractions that encode the relative node size of the documents and the node relevancy of the documents. Similar abstractions of tree nodes have been found in the SpaceTree visualization technique (Plaisant et al., 2002). This system aggregates subtrees into single triangles that represent the number nodes and the height of a subtree.

The present system also offers dynamic query capabilities first presented by Ahlberg and Shneiderman (1994). Dynamic query systems offer interface widgets such as sliders to formulate queries that dynamically impact the visualization of the result set. The HEX system offers dynamic queries on the set of retrieved results to drill down relevant subsets.

The interactive interface of the HEX system takes advantage of the Jazz Toolkit (Bederson et al, 2000) to provide pan and zoom navigational capabilities. The Jazz approach involves the building of a scene-graph of nodes and positioning camera views of the scene, an approach typically seen in 3-D toolkits. This approach offers users the focus + context view much hailed in the information visualization community (e.g. Shneiderman, 1996).

In addition to the visual navigation techniques, the current system also takes advantage of the TreeMap visualization technique (Shniederman, 1992) to organize the intra-document results. TreeMaps provide a compact representation of hierarchical data and an intuitive method for visualizing weight or rank statistics.

2.2 XML Information Retrieval

Both the XXL (Theobald and Wiekum, 2000) and the XIRQL (Fuhr and Grossjohann, 2000) offer IR statistics and ranked retrieval to XML search. The former focuses on adding similarity operator '~' that expands the query to allow terms that are similar to terms. It also provides for the '#' operator as a placeholder for zero or more element nodes. For example, the query (zoos.#.~region.Canada ~tiger) would match (zoos.public.country.Canada lion) if 'lion' was deemed similar to 'tiger' and 'country' to 'region'. Like the HEX system presented here, the operators '~' and '#' in the XXL system account for both heterogeneity of data and lack of schema, however the XXL similarity computations cannot account for many arbitrary differences in structure (e.g. /BOOK/PUBLISHER/AUTHOR vs. /AUTHOR/PULISHER/BOOK.

The XIRQL system focuses on adapting IR ranking statistics (i.e. tf*idf) to fit hierarchical data. They propose computing stats for only specially selected *index elements*. The authors also propose a weighting scheme for finding the proper level of specificity of result to return to the user if no structure is provided in the query. They use augmentation weights to calculate the influence on lower index matches with higher indices. For example, if a query searching for 'lion' matches several sibling *index elements* then this will influence the weighting of the parent *index element* and thus the parent may be returned with a higher rank. The present system uses the TreeMap visualization to represent result specificity and structure augmentation.

The XIRQL system also provides predicates that match and provide weights stats for terms across data types, vague equality, or inequalities. It does not provide the ability match based on similar structure and also requires the specification of an extended DTD and thus may not be useful in large heterogeneous environments.

2.3 Similarity

The similarity between documents in HEX is computed using the vector space model (McGill, 1983). The vector space model has been applied to cluster vectors of keywords in HTML documents for Web IR tasks (Leuski, 2000; Hearst and Pederson, 1996) but not previously been applied to the structure of XML documents. Intuitively, the application of similarity between document structure elements is more reasonable than the previous use of content keywords as the structure elements are purposefully chosen to represent the data that they contain; however, user studies are needed to validate this conclusion. As described above, the HEX system uses multidimensional scaling to reduce the dimensionality to 2 dimensions so that the similarity can be presented visually.

3.0 The HEX System

3.1 The HEX Data Model

Element Index

For each element across all documents an entry in the Element index will hold all NodeIds that hold the corresponding element name. This enables quick access to all documents (or Nodes) containing a specified element – an important operation in the HEX system. For example, to find all documents in the collection containing the element AUTHOR, a single index lookup will be required. In the present prototype system the Element Index is offered to the user in the user interface, however, in the case of large data collections this interface would become unwieldy. Future research will investigate how to organize or cluster the index for user interface in large collections.

Keyword Index

Similar to the Element Index, for each Keyword in the collection, an entry in the Keyword Index will hold the DocId/NodeIds/TermFreq that contain the corresponding keyword. The TermFreq score is used to weight the result for ranking.

Document Location Index

For each NodeId, a URL location is entered in the Location Index to find the content if requested.

Element Freq Vector

For each document, a flattened Vector of Element Names and ElemFreq weights will be stored to allow efficient computation of document similarity. The ElemFreq weight (as well as the TermFreq in the Keyword Index) can simply be the number of occurrences of the element in the document or can be weightings that have shown success in standard Web document IR research such as Okapi's tf score (Robertson et al., 1996):

- (tf = tf/[tf +0.5 +1.5*[docLen/aveLen]])

3.2 Visual Interface

Inter-Document Results

All documents in the currently retrieved collection are represented by circles with diameters proportional to the number of element nodes in the document (with a minimum and maximum size to ensure visibility). The circles are visually organized based on element vector similarity (See Figure 3). In the current implementation the similarity is calculated using a multidimensional scaling approach so that the dimensionality of the similarity distance can be reduced to two dimensions for graphical presentation (Morrison et al., 2002). An alternative would be to compute full dimensional similarity distance and present the data in categories (Hearst and Pederson, 1996; Leuski and Croft, 1996). To reiterate the point, because the similarity is calculated based on the flat set of element names, heterogeneous structured documents that contain similar data will be clustered together in the display.

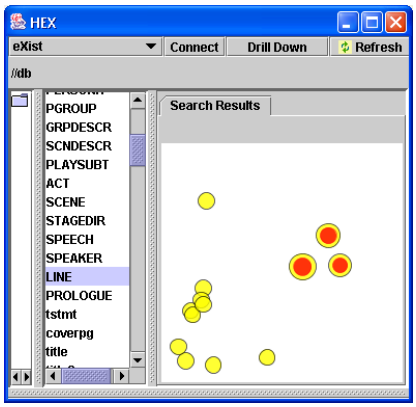


Figure 3.

The document icons are displayed within a zoomable interface (Benderson et al., 2000) to allow the user to interactively explore the document similarity.

Dynamic Queries

Documents that match the query that are in the currently displayed collection will enclose red circles that mark the match. The diameter of the red circle corresponds to the number of element nodes that match the query. The resulting icon represents the proportion element nodes in the document match the query. For example, figure 4 shows 3 documents that contain a large proportion of nodes that match the query while the other two contain no matches.

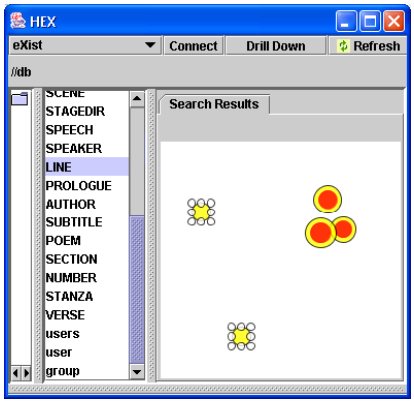


Figure 4.

Drill Down

The search operations described below operate only on the collection of documents retained after the most recently execution of the Drill-Down operation. The Drill-Down operation retains only the documents that either match the most recent query or are currently selected by the user. The second criterion (selected) allows users to retain certain documents by selecting them in with the mouse pointer even if they did not match the most recent query. The similarity distance between the documents is recomputed and the icons are translated to the new positions. Figure 4 shows the retained results from the Drill-Down operation on Figure 3 after the user selects a couple more documents.

Intra-Document Results

If the user chooses a document (or node) from the currently returned query results, HEX will present the user a visual representation of the resulting tree using Shneiderman's (1992) TreeMap visualization technique shown in Figure 5. The tree contains only the nodes in the document that match the query as well as any ancestor nodes. An alternative would be to build only the smallest subtree that contains all of the result nodes within each document, but adding the full path to the root of the tree provides useful context for the user and requires minimal processing. In the visualization the weight (rank) of the query match is represented by both the brightness of the colour and by the size of the cell. The TreeMap is the interface that allows access to the actual content of the documents/nodes. If the user chooses a node (mouse double click) then the XML text content will appear in a text browser (Figure 6).

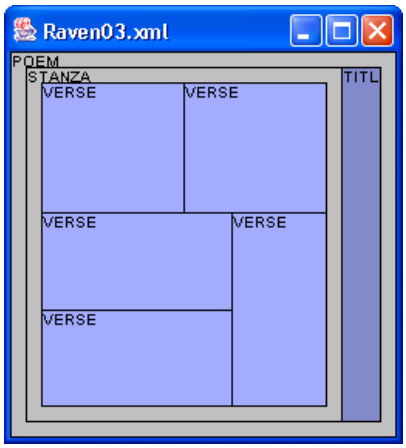


Figure 5.

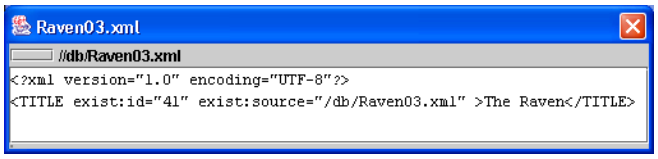


Figure 6.

The content box shown after the user chooses the TITLE element node from the TreeMap in Figure 5.

Ranking
On top of the standard IR ranking based on term frequency and inverse document frequency (not currently implemented) the HEX system weights results by the number of ancestor elements that appear in the users Element Index query (described below). For example, given two nodes /BOOK/PUBLISHER/AUTHOR/FIRSTNAME/x and /BOOK/AUTHOR/y, if the user searched for BOOK:PUBLISHER:FIRSTNAME, the node x would be ranked higher than y because it matches 3 query element names compared to 2 for y. Future research will focus on how to best integrate this weighting with the IR stats.

3.3 Search Operations
Since the Hex system is intended for large document collections, the users will most likely have a large variance in skill for developing queries. The basic operations of HEX are kept simple and intuitive. To effectively search through large data collections, the search operations are intended to be used as part of an iterative drill-down process.

Retrieval Mode
The user can choose to search in Document Mode in the case that full documents are to be returned by the query or Node Mode if all node matches are to be returned

Element Search (Match Any)
The element search allows the user to enter a list of Element names to search for. This operation has *match-any* semantics (union) so that any document (or node) that contains any one or more of the elements in the list will be returned in the result. The element search is the key to HEX's ability to excel with heterogeneous collections as it allows searching based on a subset of the element names without knowing or caring about the overall structure of the data. For example you can search for AUTHOR and BOOK without knowing about any document schema or even their hierarchical relationship to each other (AUTHOR/BOOK or BOOK/AUTHOR or AUTHOR/PUBLISHER/BOOK are all treated equal).

Keyword Search (Match All)
The keyword search allows the user to search for documents (or nodes) that have nodes that contain all of the keywords listed by the user. Users that are comfortable with today's popular Web search engine should find the *match all* semantics (intersection) of the keyword search intuitive. When issuing a query that contains both a list of elements and list of keywords the system will return all documents (or nodes) that contain all of the keywords in a node that lies under one or more of the elements in the element query

Other Search Operations
The Element Search (Match Any), the Keyword Search (Match all) and their combination are the fundamental search operations for the Hex system. Similar to popular Web search engines today, further operations (such as Element Search (Match All) and Keyword Search (Match Any)) could be offered in an advanced version of the interface.

4.0 Evaluation
The current prototype offers an intuitive, visual environment to drill down query answers in large collections of heterogeneous XML documents.

Although the speed of the query evaluation is not overly obtrusive, the current implementation has two speed bottlenecks. The first is an artifact of the prototype system

since the necessary data structures for query computation and similarity calculation need to be built at run-time. This time delay should be completely removed in a system containing a preprocessor (crawler) to fill the data structures described in section 3.1.

The second bottleneck occurs during the similarity processing. The current calculation surpasses user patience (20 seconds) if the result set is larger than 100 documents (with element vector lengths ~50). The current approach using the spring model multidimensional scaling (MDS) is O(N2) but new research in MDS suggests that this can be reduced to O(N root(N)) with an approach that is at least as effective (Morrison et al.,2002). This has proven to be approximately 3 times faster in test implementations and should be able to calculate similarity for any reasonable result set size in a reasonable time. Further studies in using a variety of test data collections are required to formalize these timings. The effectiveness of the similarity calculations can also be investigated by comparing the similarity distances calculated using the MDS approaches with the full dimensional distance.

Once similarity is calculated on a particular drilled down document set and the resulting set if organized on screen, queries of the result set can be computed with no noticeable delay.

5.0 Conclusions

This paper presented HEX, a search interface that applies research taken from information visualization and traditional IR techniques to the XML IR domain. The first prototype provides evidence that the approach may provide a useful method to effectively search through large collections of heterogeneous documents. To counter-act the loss of precision by flattening out the element structure, results are organized by element weight similarity, which provides an intuitive organization of results for the user. Extensions to the current prototype implementation are described in the next section. Further evaluation and user testing is also required.

6.0 Future Work

The prototype system is implemented as an interface to the eXist Native XML Database. To be effective for large collections, the data structures should be filled during a preprocessing stage using “crawler” software to scan the Web for XML documents similar to today’s popular search engines. It would be trivial to fill the structures outlined in the Section 3.1, but one interesting area of research would be to investigate what could be precomputed to improve the similarity stage. For example, could similarity for documents that commonly appear together in a search be precomputed?

Currently, dynamic queries are only offered for the inter-document results by highlighting the documents that match the current query. This can be extended to allow the user dynamic filtering of the nodes presented in the intra-document TreeMap representation. With this addition, the user would be able to adjust the element and keyword queries and view the effect on the TreeMap in real-time.

As mentioned earlier, the element index that is currently provided to the user in the interface would become unwieldy in large collections. One simple solution would be to simply remove the Element Index from the interface and require that the user enter the element names in a blank text field similar to the keyword search. However, the Element index valuable in providing meta-information as to the content of the collection, as well as guidance as to which terms would be useful in a search. Therefore, research into how to

best aggregate the Index for the interface should improve both the user interface and the retrieved results. The work by Theobald and Weikum, 2000, in which the element names are organized by similarity in an ontology using the WordNet open thesaurus may provide some guidance in this area.

Also mentioned in the previous sections, the efficiency of the dimension reducing MDS similarity computation can be improved and various other full dimensional similarity techniques could be explored.

Finally, the interface could be polished by adding a history of previous queries and allowing combinations of multiple queries using intersection or union. Furthermore, the various result windows (i.e. the collection of documents, the TreeMap document visualization, and the text content browser) could be integrated so that the zoomable interface could be used to navigate through the intra-document nodes and the text content instead of the additional windows offered in the current implementation.

Appendix A

Implementation References

The development of the HEX system involved integrating code and ideas from a variety of open source systems:

- The native XML database eXist (http://exist-db.org/) was used as the test XML storage upon which the HEX system was built.
- Components from the XMLdbGUI system was used for XML parsing (http://titanium.dstc.edu.au/xml/xmldbgui/index.shtml)
- Parts from an open source system fsmvis (http://www.dcs.gla.ac.uk/~morrisaj) was used to provide the multidimensonal scaling and fast multidimensional scaling engine for HEX.
- The Piccolo Zoomable Interface library (www.cs.umd.edu/hcil/jazz) was integrated to offer the graphical exploration of the query results
- The Intra-document results are presented using the TreeMap visualization technique first presented by Shneiderman in 1991, and implemented at (http://treemap.sourceforge.net)

References

C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays, In Proceedings of SIGCHI 1994, pages 313-317.

S. Amer-Yahia, S. Cho, and D. Srivastava. Tree Pattern Relaxation, In EDBT, Prague, Czech Republic, Mar. 2002.

Bederson, B. B., Meyer, J., & Good, L. (2000). Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java, In Proceedings of User Interface and Software Technology (UIST 2000) ACM Press, (in press). http://citeseer.nj.nec.com/bederson00jazz.html

N. Fuhr and K. Grossjohann. XIRQL - An Extension of XQL for Information Retrieval, In ACM SIGIR, New Orleans, LA, Sep. 2001.

P. Ganesan, H. Garcia-Molina, and J. Widom. Exploiting Hierarchical Domain Structure to Compute Similarity. Technical report, Stanford Computer Science Dept. 2001-27, Jun. 2001. http://dbpubs.stanford.edu/pub/2001-27.

Hearst and Pederson. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In Proceedings of ACM SIGIR, pages 76-84, August 1996.

IEEE CS Data Engineering Bulletin Vol. 25 No. 1, Special Issue on Organizing and Discovering the Semantic Web, March 2002.

H. V. Jagadish, L. V. S. Lakshmanan, D. Srivastava, and K. Thompson. TAX: A Tree Algebra for XML, In Int'l Workshop on Data Bases and Programming Languages (DBPL), Frascati, Rome, Sep. 2001.

A. Leuski and J. Allan. Lighthouse: Showing the way to relevant information. In Proceedings of IEEE Symposium on Information Visualization 2000 (InfoVis 2000), pages 125-130, 2000.

A. Leuski and B. Croft. An evaluation of techniques for clustering search results. Technical Report IR-76. Department of Computer Science, University of Massachusetts, Amherst, 1996.

M. J. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.

W. Meier. eXist: An Open Source Native XML Database. In: Akmal B. Chaudri, Mario Jeckle, Erhard Rahm, Rainer Unland (Eds.): Web, Web-Services, and Database Systems. NODe 2002 Web- and Database-Related Workshops, Erfurt, Germany, October 2002. Springer LNCS Series, 2593.

A. Morrison, G. Ross, and M. Chalmers, Achieving Sub-quadratic Multidimensional Scaling through the Combination of Sampling, Clustering and Layout Algorithms, in Proc. IEEE InfoVis 2002, Boston, Massachussets, USA, October 28-29, 2002. IEEE Computer Society Press. http://www.dcs.gla.ac.uk/~matthew/papers/SubQuadMDS.pdf

A. Nierman and H. V. Jagadish. Evaluating Structural Similarity in XML Documents, In Int'l Workshop on the Web and Databases (WebDB), Madison,WI, Jun. 2002.

Northern Light. http://ww.northernlight.com.

C. Plaisant, J. Grosjean, and B. Bederson. SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation, In Proceeding of InfoVis 2002.

S. E. Robertson, S. Walker, M. M. HancockBeaulieu, M. Gatford, and A. Payne. Okapi at TREC-4, In D. K. Harman, editor, The Fourth Text REtrieval Conference (TREC-4), pages 73--96, Gaithersburg, MD, 1996. National Institute of Standards and Technology, Special Publication 500-236.

D. Roussinov and H. Chen. Information Navigation on the Web by Clustering and Summarizing Query Results, In Information Processing and Management 37 (6) (2001) 786-816.

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. Information Processing and Management, 24(5):513{523, 1988.

Shneiderman B. The eyes have it: A task by data type taxonomy for information visualizations, In Proceedings IEEE Visual Languages, pages 336-343, Boulder, CO, Sept 1996. http://citeseer.nj.nec.com/shneiderman96eyes.html

B. Shneiderman. Tree visualization with tree-maps: A 2-d space-filling approach. ACM Transactions on Graphics 11, 1 (1992), 92-99.
http://citeseer.nj.nec.com/shneiderman91tree.html

Srivastava, S. Al-Khalifa, H.V. Jagadish, N. Koudas, J. M. Patel, and Y. Wu. Structural Joins: A Primitive for Efficient XML Query Pattern Matching, In Proceedings of the ICDE Conference, 2002.
Theobald and G. Weikum. Adding Relevance to XML, Int'l Workshop on the Web and Databases (WebDB), Dallas, TX, May 2000

J. Wise, J. Thomas, K. Pennock, D. Lantrip, M. Pottier, and A. Schur. Visualizing the non visual: Spatial analysis and interaction with information from text documents, In Proceedings of IEEE Information Visualization, pages 51-58, 1995