

12-2003

## Visualization of Internet Web pages based on authority and word frequency

David Navarro  
*University of Texas-Pan American*

Follow this and additional works at: [https://scholarworks.utrgv.edu/leg\\_etd](https://scholarworks.utrgv.edu/leg_etd)



Part of the [Computer Sciences Commons](#)

---

### Recommended Citation

Navarro, David, "Visualization of Internet Web pages based on authority and word frequency" (2003).  
*Theses and Dissertations - UTB/UTPA*. 610.  
[https://scholarworks.utrgv.edu/leg\\_etd/610](https://scholarworks.utrgv.edu/leg_etd/610)

This Thesis is brought to you for free and open access by ScholarWorks @ UTRGV. It has been accepted for inclusion in Theses and Dissertations - UTB/UTPA by an authorized administrator of ScholarWorks @ UTRGV. For more information, please contact [justin.white@utrgv.edu](mailto:justin.white@utrgv.edu), [william.flores01@utrgv.edu](mailto:william.flores01@utrgv.edu).

**VISUALIZATION OF INTERNET WEB PAGES  
BASED ON AUTHORITY AND WORD FREQUENCY**

**A Thesis**

**by**

**David Navarro**

**Submitted to the Graduate School of the**

**University of Texas – Pan American**

**In partial fulfillment of the requirements for the degree of**

**MASTER OF SCIENCE**

**December 2003**

**Major Subject: Computer Science**


**VISUALIZATION OF INTERNET WEB PAGES  
BASED ON AUTHORITY AND WORD FREQUENCY**

A Thesis

by

David Navarro

Approved as to style and content by:



---

Richard H. Fowler, Ph. D., Professor  
Chair of Committee



---

Zhixiang Chen, Ph. D., Associate Professor  
Committee Member



---

Wendy A. Lawrence-Fowler, Ph. D., Professor  
Committee Member

December 2003

## ABSTRACT

Navarro, David, Visualization of Internet Web Pages based on Authority and Word Frequency. Master of Science (MS), December, 2003 43 pp., 6 figures, references, 16 titles.

The growth, accessibility, and integration of the World Wide Web with contemporary information utilization provides a rich domain in which to explore information retrieval systems. One approach in the evolution of retrieval systems couples successful and long- standing techniques of information retrieval with new techniques, such as visualization. The system developed and reported in this thesis takes this approach. It builds upon well-known techniques of information retrieval including stemming, keyword matching, and cosine similarity. It also incorporates the new and relatively successful hubs and authority approach, which describes Web documents by their reference by other documents. Finally, it develops a new and unique approach to document visualization that encodes these metrics in a single visual representation. This new, easily scannable representation, allows the user to interact with search results as the scope of search is expanded dynamically across the Web.

## ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Richard H. Fowler, for his guidance, help, encouragement and most of all his patience. I also thank Dr. Wendy A. Lawrence-Fowler and Dr. Zhixiang Chen, who along with Dr. Fowler served on my thesis committee. A thanks also goes to Yavuz Tor for answering several questions used in the implementation of the *IRV* system. Finally, I would like to thank The University of Texas – Pan American Computing and Information Technology Center and NASA Grant NAG 9-1169 for their support.

## TABLE OF CONTENTS

ABSTRACT .....	iii
ACKNOWLEDGEMENTS .....	iv
TABLE OF CONTENTS .....	v
LIST OF FIGURES.....	vii
INTRODUCTION.....	1
RELATED WORK .....	4
STATEMENT OF THE PROBLEM .....	11
SYSTEM .....	12
1.1 Overview .....	12
1.2 IRV System Architecture .....	17
1.3 IRV Visual Structures .....	20
1.3.1 X-axis/Y-axis System.....	20
1.3.2 P/D System.....	21
1.3.3 Example of Search and Visual Page Structure.....	22
1.4 System Implementation.....	25
1.4.1 CDynLookDoc / CDynLookView Classes .....	25
1.4.2 Utility (Util) Classes.....	26

1.4.3 Support Classes .....	29
FUTURE WORK .....	32
CONCLUSIONS .....	33
REFERENCES .....	34

## LIST OF FIGURES

Figure 1. Results of a search using Information Retrieval Visualization (IRV) system...	13
Figure 2. System data and control structure for data retrieval, analysis and display.....	15
Figure 3. Visual representation of a single page.....	20
Figure 4. User entered a keyword set describing information need and a starting page..	21
Figure 5. A second document has been retrieved and its visual representation displayed.	22
Figure 6. Example set of visual document representations formed as part of a search....	22



## INTRODUCTION

With the Internet still continuing its explosive growth, there is an overwhelming number of documents (web pages) that range from poor quality and entirely useless to high quality and having the exact information needed. Of course, we are assuming somewhere on the Internet on some server the “perfect” document that holds the information needed exists. The group of researchers specializing in information retrieval deal with several different obstacles. What makes a hard job even harder is that when these researchers have to deal with documents that have no formally defined structure, including the typical Internet web page. These information retrieval systems have to find appropriate documents based on whatever attributes they can extract from these documents.

Various techniques are used to determine relevance, such as searching by keywords (Ng and Zheng, 2001). This technique alone does not work very well, since the query on a keyword or list of keywords can return a large number of documents. Ranking web pages using terms can be applied to decrease the high number of returned web sites. While ranking web pages is not an exact science, some approaches are well accepted, including the technique or algorithm used in the programming portion of this paper that was used for testing. Another thing to note about keywords is that the keyword itself can

submits a query on *automobile*, the computer will not consider the word *automobiles* as match and will keep the keyword match number the same.

To combat the problem, with all suffixes not just the –s example, an algorithm is applied to both the keyword or keyword(s) and all the documents on the page. This algorithm converts each term in the document to a “stem” (Porter, 1980), and that stemmed version is used in the keyword matching. An additional approach to try and determine relevancy is called *similarity-based retrieval*. In this approach a user gives an information retrieval system a document *A*. Document *A*’s similarity to every document in the system is defined as the number of common terms. A problem with similarity-based retrieval, as with all other retrieval systems, is that the set of documents returned is usually still large. At this point options are given to the user, such as letting the user choose the higher relevant documents and using those (document *A* and the first relevant document) as input into the system. One last concern is synonyms and homonyms. Using keywords to find what we consider relevant is very powerful and intuitive, but to a computer system, if you ask for information on “stones” it will not consider “rocks” relevant. A straightforward way to overcome this problem is using a synonyms look up dictionary. This does have the potential of causing its own problems. Using synonyms to extend queries might result in those added words having slightly different meanings, and thus degrading the quality of query results. The opposite is also true. If the user is doing a science project and doing research on stars, an information retrieval system that returns a list of the current Oscar nominees is not exactly what he/she wanted.

The work described in this thesis designed and implemented the Information Retrieval Viewer (IRV), a tool to help view what documents are relevant to a user's query, using well understood techniques of information retrieval, together with new visual representations of search results. The IRV provides functionalities to the user of any type of electronic document retrieval system retrieving relevant documents based on a few of the accepted standard approaches, and how "good" or "accurate" each result of that approach performed is. IRV shows in a simple to read and intrepid plot graph representations of which documents are most relative based on a single approach, such as using key words or using the idea of authorities, and show the return using both together. Built on similar previous research (Harmon, 1987) and usage of code we also have the ability to find specifically which paragraph in the document has the highest relevance value. Significantly, these relatively well understood approaches of information retrieval are combined in a system that visually displays search results to user in a form that simultaneously encodes information about several aspects of the retrieved documents in an easily and rapidly scannable form.

The next section examines past work relevant to the project.

## RELATED WORK

*Empirical evaluation of information visualizations: An introduction.* Noting the importance of improved methods in task analysis, usability evaluation, usage analysis and visualization-intrinsic user interfaces, the authors (Chen and Czerwinski, 1981) attempt to improve upon the field of information visualization through an empirical evaluation of the latter by evaluating VIBE, Treemap and Cone Trees. Beginning with an evaluation of an information visualization known as VIBE, the authors describe its method of visualizing as represented spatially and a method of testing situations beginning with the simplest instances and moving to more complex situations. Further illustrating the use of empirical evaluations with Treemap, the author describes space-filling visualizations that were created with end users in mind. With Cone Trees the author describes the need of empirical evaluation in the implementation of hierarchies in data structures in information vacillation. Ending with a summary of three different articles focusing on the development of information visualization, the author describes the lack of user-centered design in information visualization despite the abundance of information on visualization techniques.

*Top Ten Problems in Visual Interfaces to Digital Libraries.* In answer to a lack of consensus to the definition a general theory of information visualization, the author of this paper (Borner and Chen, 2002) chose to outline a list of the ten most salient

questions faced by researchers in this field. Beginning with a list of top ten issues related to the field of visual interfaces of digital libraries, the authors mention problems faced in the fields of computer graphics, information retrieval, and concerns with visualizations in general. Followed by the actual top ten list of problems the field of visual interfaces of digital libraries, the authors attempt to set a groundwork for the establishment a focus and guidelines to this area of research.

*Navigating Cyberspace with CyberGeo Maps.* L.E. Homquist, H, Fagrell and R. Bus (1997) have studied the importance of CyberGeo maps. These CyberGeo Maps are used to inform the viewers about the age, size and directory structure of website and website domain. Maps like these will help people to understand the many fields and areas that the Internet or cyberspace has to offer. The two examples that are used in this study start with a Swedish newspaper website and continue with the website of the Association of Computing Machinery (ACM). They help show the development of the World Wide Web (WWW) and foreshadow their beliefs of what is to come.

*Viewing Stemming as Recall Enchantment.* Much research has been done on stemming from analysis on language for development in algorithms such as Porter's and others. These researchers (Kraaji and Pohlmann, 1966) take an objective approach on how stemming or suffix stripping results and determine whether it us an advantage or disadvantage in performance. In this study, the focus was on the recall measurements using linguistic and nonlinguistic stemmers. The results proved that recall was significantly better and had an advance in precision when using a linguistic stemmer.

This would definitely help users, to retrieve specific and higher quality information from information retrieval systems.

*A Failure Analysis on the Limitations of suffixing in an Online Environment.* This study is very similar to “Viewing Stemming as Recall Enactment” with the exception that researchers are in an online environment, and the testing these researchers do are not categorized by linguistic and nonlinguistic stemming. Here, Harmen focuses on the importance of finding the most effective and quantitative data retrieved by different suffixing algorithms the three programs studied here were Cranfield 1400, Medlars, and CACM collections. Each were tested and evaluated based of word variants and selective stemming from requested information because finding the most successful technique offered the to the public would help gain more users the researchers added.

*Visualizing websites using a hierarchical table of Contents Browser: WEBTOC.* David A. Nation and his colleagues (Nation et al., 1997) have developed a method to help users navigate through different websites. This type of navigation contains a list of a hieratical table offering information about the site and its geographical data with in the Internet. The authors demonstrate this method with a software program called WEBTOC. Using WEBTOC users can quickly view through the branches of the system or website domain and find any information the need. Navigation will be quick and easier for the viewer or user to understand.

*Does “Authority” mean Quality? Predicting Expert Quality ratings of Web Documents.* In this article AT&T Researchers tested the linking structure of the World

Wide Web (WWW) with the idea of “Authority and Hubs” to determine if the notion of Authorities does actually produce quality results. They compared the results determined by Authorities vs. those a human would call the “best page”. The researchers (Amento, Terveen and Hill, 2000) used a dataset with 0.32 of all documents were considered “high quality”. Overall the researchers noted that they were surprised how well both the content-based metric performed and even more so the link-based metric or the notion of “authorizity.”

*Stable Algorithms for Links Analysis.* In this paper, after publishing one article on the topic, the researchers further analyze both the Kleinberg HITS and Google PageRank are eigen vector methods for identifying Authority Pages. Yet they don’t always produce consistent answers. The authors develop two algorithms which with empirical data prove to give stable results (Ng, Zheng and Jordan, 2001).

*Do thumbnail previews help users make better relevance decisions about search results?* Researchers at Microsoft explore different ways of providing the same information with different interfaces (Oziadosz and Chandrasekren, 2002). The first way they offered the information to the user was in a pure textual interface. The text was formatted so the user could find what was more important with bold font, etc. so it was not just a string of text. The second was a pure graphical way of navigating and did not use any text. The final way the data was presented as a navigation scheme to the tested users was by a combination of the two a graphic with text to the right with clips of the text found on the linked web page.

*PageRank, HITS, and a Unified Framework for Link Analysis.* This paper goes over the two most popular algorithms used to determine Authorities and hubs values, the HITS and PageRank algorithms. It discusses the difference between the two (Ding, 2002). The authors then generalize all the key concepts, and combine them into their “Unified Framework”. It concludes with two sets of empirical data that shows both HITS and PageRank are highly correlated with indegree rankings.

*The Semantic space of Online Course Descriptions.* This paper discusses an example of Information Retrieval Visualization system (Borner, 2000). The problem lies in how online course descriptions hold valuable information. Using the text value of the descriptions, a semantic value, and placing them or plotting them on a two-dimensional graph gave insight of how relative two classes that are not even in the same department can be related to each other. The article also gave in addition a three-dimensional topological map of courses offered.

*An algorithm for suffix stripping.* Suffix stripping is a widely used algorithm that is not only very effective, but also straightforward to implement and runs surprisingly fast. The paper itself explains why it works and shows step-by-step how to implement it. Porter (1980), the author, discusses about its usefulness to information retrieval. It illustrates examples of words with the same root, and shows that once the suffix is stripped, they are the same for information retrieval purposes. He also states how when dealing with large amounts of data, like in a typical information retrieval system, keeping the size and complexity smaller is “always advantageous.” One example of how different



words have the same value compares “connect”, “connected”, “connecting”, “connection”, and “connections”. Again, for information retrieval all mean the same thing. He concluded by discussing effectiveness and how he measured it. Porter used a vocabulary of 10,000 words with this algorithm, and it reduced the number of words (stems) to 6,070, approximately one-third.

*ACHIA: Automatic Construction of Hypertexts for Information Retrieval*

*Applications.* ACHIA is another attempt to automate authoring and constructing of hypertext (Maristella, Crestani and Melucci, 1995). The authors’ state that the main problems are:

1) Capability of managing different types of objects; an OODBMS is adopted for storing different types of objects that need to be managed in a digital library, 2) Integration of different retrieval approaches, such as query and browsing: the result of such integration enables the user to actively interact with the system, 3) Capabilities of interfacing heterogeneous data sources that have to be completely accessible through a standard web client, and 4) Implementation of relevance feedback techniques using one of the spreading activation methods. The paper also detailed information about the architecture being designed with distributed objects connecting different types of software tools both on client and server.

*Authoritative Sources in a Hyperlinked Environment.* This is another paper that is widely used and is considered a clever idea on how it works (Kleinberg, 1998). This is the original paper that introduces the notion of *authority* and *hubs*. It gives a circular

definition for each, which is often criticized. An authority is a document such as a web page that holds high quality data and it has a high count of in-degrees from hubs. A hub is a document, or once again a web page, that contains a high count of out degrees to authorizes. The paper also gives detailed information and empirical data on its value in information retrieval.

*Exploiting Hyperlinks for Automatic Information Discovery on the WWW.* This (Chen, Hsu and Hou, 1998) is another paper on how to use hyperlinks to discover what is important and give different values to documents on the Internet. The paper discusses the design of site traversing and proposes a new evaluation method.

*Using the Abstract Text Viewer.* This paper (Fowler, Tor and Navarro, 2003) is our previous work and uses various ideas on text value and information visualization. The work on word frequency specifically directly relates and the algorithm for `HTMLParser` class is used for the current research project.

## STATEMENT OF THE PROBLEM

Information access is a need that has always existed. Since the advent of computers, various techniques have been developed to exploit processing capability by automating aspects of human directed search, as well as develop techniques that can only be performed by computers. Within the last decade the growth, accessibility, and integration of the World Wide Web with contemporary information utilization provides a rich domain in which to explore information retrieval systems that couple long standing techniques of information retrieval with newly evolving techniques in novel ways such as visualization. The Information Retrieval Visualization program (IRV) developed and reported in this thesis takes that approach. It builds upon well-known techniques of information retrieval including stemming, keyword matching, and cosine similarity. It also incorporates the new and relatively successful hubs and authority approach, which describes Web documents by their reference by other documents. Finally, it develops a new and unique approach to document visualization that encodes these metrics in a single visual representation. This new, easily scannable representation, allows users to interact with search results as the scope of search is expanded dynamically across the Web.

## SYSTEM

### 1.1 Overview

The Information Retrieval Visualization program (IRV) is a software system, designed to help a user digest the often overwhelming, amount of data provided by standard web based information retrieval systems. IRV uses several industry standard techniques to provide output including: stemming, authority and hubs, and keyword matching. The IRV provides functionalities to the user of any type of electronic document retrieval system of what documents are relative based on a few of the accepted standard approaches and how “good” or “accurate” each result of that approach performed. The system determines what information is useful and where it is located. IRV shows in a simple to read and interpret visual representation what document was most relative based on a single approach such as using key words or using the idea of authorities and shows a metric derived from their combination. Built on similar previous research (Harman, 1987) and usage of code the system also finds the specific paragraph in the document of highest relevance.

IRV is based on keyword search techniques, page authority, and paragraph identification. IRV’s keyword based facilities utilize a basic search of stemmed keywords. A user enters keywords, and those keywords are searched for throughout the document (web page). The results of keyword matches are entered into a storage location that associates a page’s URL with the number of matched keywords. The system also

utilizes page authority value in retrieval. Authority value in IRV is defined as the number of hyperlink references, i.e., “href” links, from a web page to a web page. For example if the system has processed web page ‘X’ and is now processing web page ‘Y’ and web page ‘Y’ has a hyper link to web page ‘X’ then the authority value on ‘X’ goes up by one. This does not change or have any kind of effect ‘Y’ authority value. Additionally, within a single document IRV determines which paragraph is most relevant to the entire document. This is done by first making a vector of all the words used in the entire document. Then, each every word in each paragraph is compared to the search keywords. Paragraphs with more keyword matches are judged more relevant.

To begin a search the user enters keywords to be matched and a starting page URL. With this input the system begins retrieving pages that are pointed to by that page, expanding the search over the web as a breadth first search. As described below, the system visually displays search results to the user, and the user controls the time of the search. The longer the IRV system runs, the more accurate it becomes since it processes more web pages and correctly computes the authority for all web pages found. In principal the program could explore the entire web. Figure 1 below shows results of a search.

Output of IRV system is intuitive and simple to read. The output is a document set graph that is refreshed or redrawn after every time a new web page document is processed, reflecting progress of the search. Thus, the results visualization window is continually changing, as individual page representations are added. Individual Web pages

are represented by a collection of lines, described below, and this single page representation is placed at a particular x, y point in the document set graph. The x-axis reflects the number of keywords found, with high keyword count pages placed near the origin. The y-axis reflects the authority value, with high authority value pages placed near the origin. The “best” pages, i.e., those with the most keywords matched and the highest authority, would be placed in the lower left corner of the document set graph.

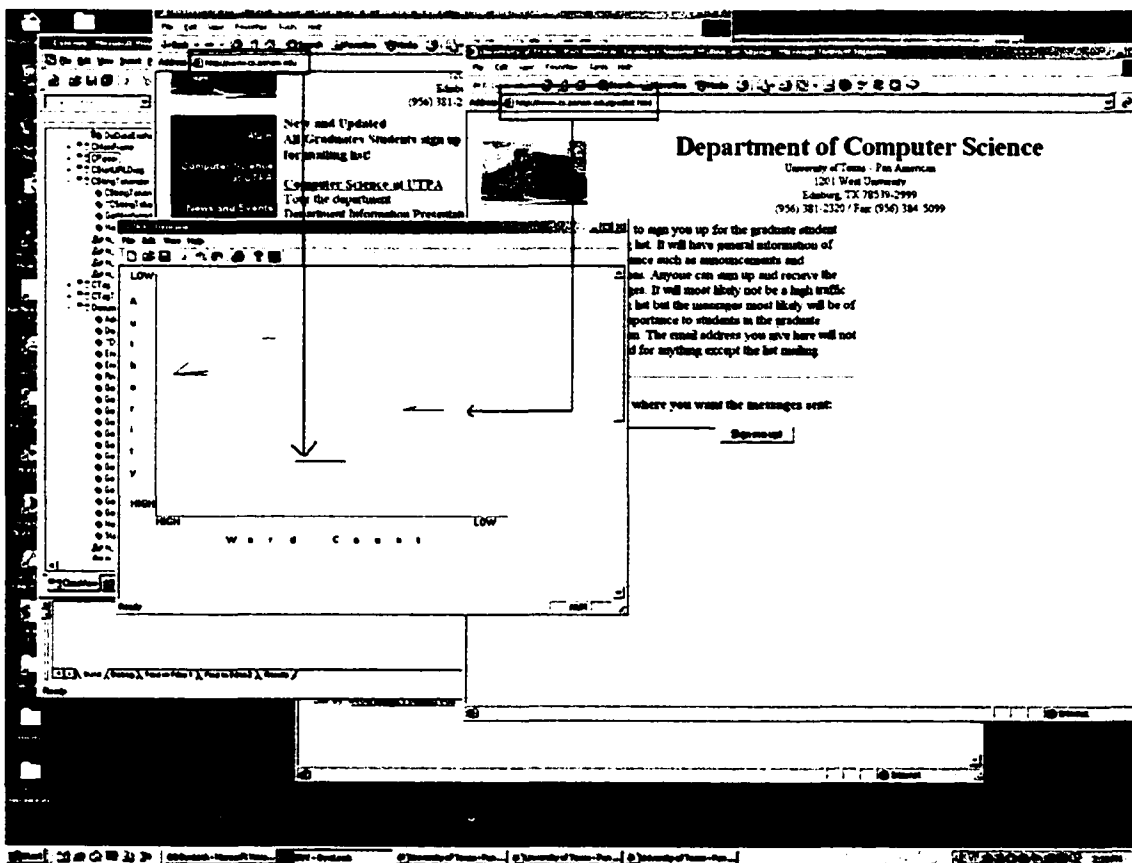


Figure 1. Results of a search using the Information Retrieval Visualization (IRV) system.

Displayed as a search progresses, visual page representations are dynamically added to the results visualization window. In the example above the page representation shown on the left on both screen and visualization window reflects a) a relatively long document, as represented by the length of the line, b) has a moderate number of the search keywords, as represented by its placement midway along the x, or keyword count, axis, and c) has a relatively high authority score, as represented by its placement at the high end of the y, or authority, axis. The page representation shown on the right is a) a shorter document, with a shorter line, b) has fewer keywords, as it is to the right of the first page, and c) is of lower authority, as it is higher. Additionally, the second document has one paragraph that is closely related to the overall content of the page. As page representations are added to the display, the range of values for authority and number of keywords, of course, changes and it is a page's relative position in the dynamically adjusted range that determines its placement.

As noted, the display element representing a single page is formed from a set of lines. It is the combination of two lines characterizing the system's extraction of information about a page's number of words, i.e., the word count line, and paragraph structure, i.e., paragraph content lines. The display element's form facilitates comparison with other pages' content. For all pages a horizontal line represents the number of words that document has, the word count, or length, line. Given that single web pages may range from a simple list of links to entire reports, this information can be quite useful to the user in determining which document to retrieve and examine.

For pages with multiple paragraphs, paragraph content lines are drawn at different angles to the horizontal line. Each of these lines represents a paragraph in the document. The angle of paragraph content lines represents the relevance individual paragraphs in the document have to the document as a whole. This provides some information about the structure of the page. It gives an indication of how “focused” the document is, as detailed in the following section. Documents that contain a group of paragraphs closely related to each other will have paragraph lines relatively close to the horizontal and close together. For groups of paragraphs that are not similar, i.e., less “focused”, the paragraph lines will spread out more and be farther from the horizontal word frequency line.

## **1.2 IRV System Architecture**

The input to get the program started is a starting Internet address (or IP address) and a query key word(s). The program proceeds in search and analysis from the starting URL, extracting page keywords, computing authority values, and displaying the visual representation to the user, as the search progresses. The user may pause or terminate the search at any point, as he explores search results. Figure 2 summarizes the search and retrieval process of the system. As shown, processing and display of the visual representation is done on the fly, or dynamically.



IRV Data Structures: {URL, Match Word Count, Authority Value, Number of Paragraphs,  
 Paragraph relevancy to Document value for each paragraph}

Read from user: Starting URL address; Keywords  
 Stem Keywords;

**while** (user does not stop...)

    Download URL web page to a temp file  
     Parse any URLs in document

**if** (new URL has been processed)

        increment authority value in DataStruct for that URL

**else** // it has not been processed

    {

        Enqueue page // for breadth first search

        Remove any scripting (ie. JavaScript, html, XML ...), leaving only page  
         text

        Stem every word in file; Remove “stop words”

        Enter URL into DataStruct

        Compare every word keep count of matched words

        Initialize Authority value to 1

    }

    Redraw visual representation window to include new data.

**end while;**

Figure 2. System data structures and control structure for data retrieval, analysis and display.

The program goes to the starting web page, parses the web page, and determines the number of matching keywords after stemming all words. Then, doing a comparison between the stemmed version of the word in the document and the stemmed keyword or

keywords given by the user. For example the word *Computer* would be stemmed into *comput* and if the document or web page had the word *computing* porter's algorithm would stem it to *comput* the program would then update the record field representing the number of keyword matches of that document by one since it would equal a match.

Authorities are determined through the linking structure on web pages (Chang, Hsu and Hou, 1998; Kleinberg, 1998; Ng, Zheng and Jordan, 2001). Once a the URL address or the IP address of the web page is given by the user at the beginning of the run, the address is entered into a data structure that holds a in-degree count which represent the *authority value* of that web page.

Program execution algorithm is fairly straightforward. The user enters two pieces of data: starting URL web page, and keywords the information retrieval system will try to return. First, the program will make a connection, download the web page to a temporary file, and stem every word in the file using porter's algorithm (Porter, 1980), which, as stated previously, is very effective (Harmon, 1987) and does improve the return (Kraaij and Pohlman, 1996). Then, after the user's search keywords are also stemmed, a linear search is done through the document, storing all URLs found in the document and a counter of the (stemmed) keyword matches.

The URLs extracted from a page serve as the basis for the breadth first search, and are then entered in a Queue (FIFO) structure. The only way that it does not enter the queue is if it's already in the queue (duplicate entry) or it has already been visited. In either case the Authority count of that URL is update to reflect the repetition of that URL found. This URL Authority count is used to determine the visual page representation position on Y-axis, which dynamically change as more web pages are processed and the range of keyword and Authority values change.

### **1.3 IRV Visual Structures**

Information Retrieval Viewer shows the user multiple dimensions of information at once. It is split into two main parts, the X-axis/Y-axis system and P(aragraph)/D(ocument) system.

#### **1.3.1 X-axis/Y-axis System**

Keyword match is a primary element in placement of the visual page representation. The words are stemmed to enhance results and keep higher counts on our values to determine effectiveness. Some current studies on authorities (Armato, Terveen and Hill, 2000) had been conducted by researches at AT&T. Authority value of a page determines placement on the Y-axis. Placement at X and Y points is relative to the range of keyword matching authority values that is constructed as pages are analyzed and added to the set. Visual representations are positioned at a point on an X/Y plane, where the X-

axis represented the matching keywords and the Y-axis represented the authority value on the document.

### 1.3.2 P/D System

While trying to give users as much information as possible without overwhelming them we implemented a way to show how relative a paragraph was to the document it was on. This method is very well known of creating a vector of terms for the entire document and creating a vector of terms on each paragraph and measuring the difference of each. This also gave us the opportunity to show an easy to understand (Holmquist et al., 1997) way of representing the length of each document and each paragraph. The result was our ability of instead of plotting generic points we plot points with this structure. This is done with the following function:

$$D(p) = \frac{R(p, d) - \frac{\sum R(p, d)}{n}}{\sqrt{\sum_{p \in d} \left[ R(p, d) - \frac{\sum R(p, d)}{n} \right]^2}}$$

Where  $R(p, d)$  is the cosine similarity of term vectors of paragraph  $p$  and the document  $d$ , and  $n$  is the number of paragraphs in the document.  $D(p')$  analyzes the number of steps that the relevance value  $R(p', d)$  is away from the mean relevance value. Each step is equal to the standard deviation of the relevancies of paragraphs to the containing document. This is a naive statistical approach (Nation, Plaisent and Marchionini, 1998) to define the relevance of the paragraph to its document, and not

intended to provide the best solution but a satisfactory one. Figure 3 below, shows the visual representation of a single page.

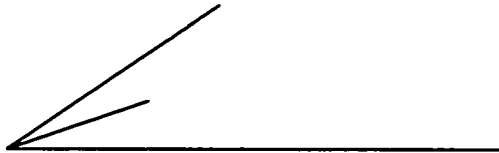


Figure 3. Visual representation of a single page. The visual representation reflects a page's length and paragraph structure. All pages are represented by a horizontal line. The length of the line represents the number of words in the page. Pages with multiple paragraphs have additional, non-horizontal, lines, each of which represents a paragraph in the document. Paragraphs with content close to that of the entire document are represented by lines with small angles from the horizontal line.

### 1.3.3 Example of Search and Visual Page Structure

As described above, search results are presented incrementally to users. After the user enters keywords to describe the content of interest and a starting page location the system begins the search. The system extracts from the starting page all pages (urls) that the page contains, retrieves these pages, and continues in this fashion, thus performing a breadth first beginning with the starting page. For each page retrieved, the single page representation is formed, reflecting page length and paragraph structure, and the representation is displayed. As each page in the search is processed, that single page representation is displayed. Note that the system continues indefinitely, depending on the

user's interaction to select individual documents from the display window to read in a browser window. The sequence of figures below shows an example search.

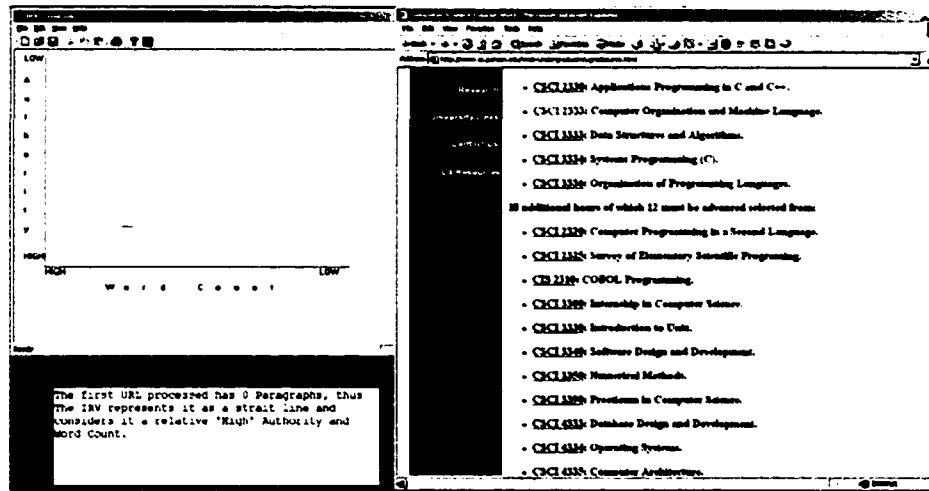


Figure 4. User entered a keyword set describing information need and a starting page.

The visual representation of the first page retrieved in the search is displayed.

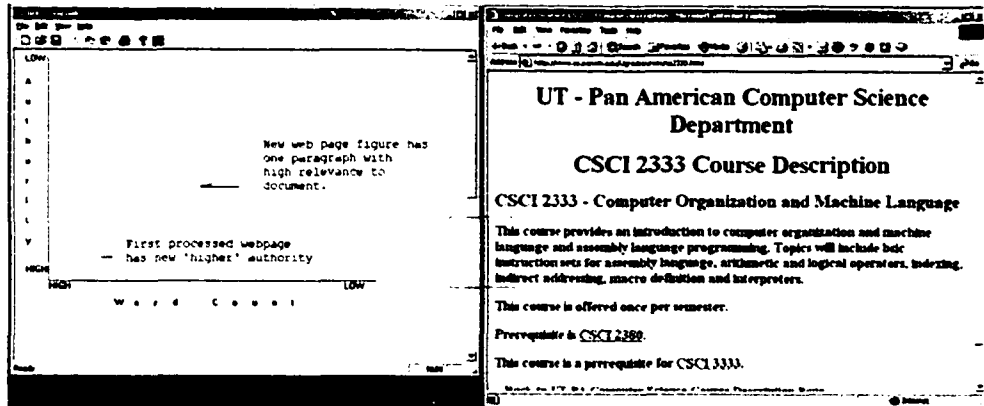


Figure 5. A second document has been retrieved and its visual representation displayed, after the system has continued its search from the first document page links.

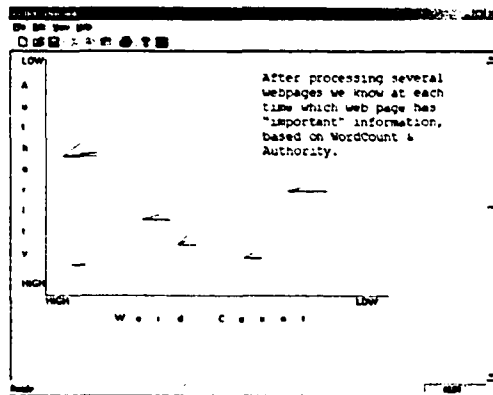


Figure 6. Example set of visual document representations formed as part of a search

## 1.4 System Implementation

IRV is based on the industry standard way of Object-Oriented Programming (OOP). The computer specifications it was built and tested on was a Dell Dual Xeon Processor with 1 GB of system ram using Microsoft Windows 2000 for an operating system and was compiled with MS Visual Studio 6.0 with service pack 5. Application programming interface used was the Microsoft Foundation Classes (MFCs), DOCUMENT/VIEW Architecture.

### 1.4.1 CDynLookDoc / CDynLookView Classes

IRV has two main classes. The first one is name CDynLookDoc, which handles the main data needed for the program, And CDynLookView, which handles all the output for the data.

CDynLookDoc has several member functions and attributes, as is a polymorph from the CDocument class from the MFC API. The class can be divided into four main parts: Attributes, Operations, Overrides, and Implementations. Attributes are variables that describe the data being processed inside the CDynLookDoc class. Current imathword, iwordcount, and iworddegrees are the attributes. Under operations section is where a debugging function "PrintData()" is prototyped. This function was used only in the testing phase. Overrides portion of the class contains the public member functions that were inherited by the CDocument class. And are all virtual functions they are:



“virtual BOOL OnNewDocument()” and “virtual void Serialize(CArchive& ar)”.

Implementations are where the ‘meat’ of the code and functionality is placed. The foremost functions are AddURLtoList(CString source, Document \* doc), start(), the destructor for the class ~CDynLookDoc(), and few more virtual debug functions “virtual void AssertValid() const;” and “virtual void Dump(CDumpContext& dc) const;”

The other class mentioned earlier of equal significances is CDynLookView. This class has the three sections as apposed to the CDynLookDoc (it does not have any operations) but handle the way the output is displayed instead of the data manipulation. CDynLookView only has one attribute it is the function “GetDocument()” that will return a CDynLookDoc pointer. This is the means of messaging the DOCUMENT class in the architecture. Overrides in the VIEW class are all virtual and are “OnDraw(CDC\* pDC)”, “PreCreateWindow(CREATESTRUCT& cs)”, “OnInitialUpdate()”, “OnPreparePrinting(CPrintInfo\* pInfo)”, “OnBeginPrinting(CDC\* pDC, CPrintInfo\* pInfo), and OnEndPrinting(CDC\* pDC, CPrintInfo\* pInfo). The implementation section contains debugging virtual functions AssertValid and Dump, it also contains the destructor ~CDynLookView().

#### 1.4.2 Utility (Util) Classes

The Util Classes are “charops”, “Document”, “HTMLParser”, “Link”, “Paragraph”, “Parser”, “porter”, “StringTokenizer”, “Tag”, and finally “TagTokenizer”.

Going through further detail about each one in alphabetical order.

The first Util class is charops. Charops stands for “character operands” and has two member functions. The first is `::IsMySpace(TCHAR c)` which returns a bool value of true if the character being passed any type of space such as a tab space, or a newline etc. The second is `::IsMyAlpha(TCHAR c)` which also returns a bool value it returns false if the char variable ‘c’ is not a letter.

The second class is Document, this is not the same as Document is the sense of Document/View architecture mentioned earlier. This is the class that handles or embodies the web page once downloaded and stored in the temp file. The main member function that this classes holds is the `::GetMatchWordCount(CString &nList)`, which returns an integer value. It passes into it, a CStringList that list contains all the keywords, stemmed of course, the user input in the beginning of the run. Another important member function in this class that is heavily used is the `::GetParRelevance(Paragraph *p)`. This function returns in a ‘double’ the value of relevance to the main document for each paragraph. Which is later used by CDynLookView to draw each paragraph line vector. The final member function that has high importance is `::GetLink(int aIndex)`. GetLink is ran several time in a to pull every Link embedded in the web page document to either update authority values or queue for further processing.

HTMLParser is the pre-processor for the Parser class. It handles setting up anything the Parser class might need, and acts as a bridge between the CDynLookDoc

class and the Parser version of the document. The most notable functions are `::ReadFromFile(CString &aFileName, CString &aContext)` and `::ReadFromInternet(CString &aURL, CString &aContext)` which will handle storing the web page into the temporary file. The system does support the original URL web page being located on the local system.

The next class is the Link class. Link a 'safe' way of holding a web page link, while it may seem easier just to create a variable `CString` to hold the URL data, by creating an entire class this allows us to create safety features for example making sure the web page still exists or if there is any problems with the link itself perhaps the anchor missed typed and only put 'http' and forgot the other 't'. The member functions of this class could prevent this from entering the URL queue of valid locations.

The Paragraph class has information dealing with relativity ( `GetDocRel(CMapStringToPtr &aVector)` )for the entire document; it also has another member function `GetWordCount()` that will return the count of all discrete words in each paragraph. "Parser" class is probably the most difficult to implement with an incredibly simple interface to it. The only important access needed from the Parser class is a member function call "`ParseDocument()`". `ParseDocument()` take the current Document class pointer and does the work of removing all scripting entrenched inside the document. All headings, image references, etc. are completely removed, too.

The Porter functions are named after the person who designed the algorithm used

to do the word stemming, M.F. Porter. The Porter functions are just functions. They are not implemented as a class but global functions. The design reason for that was based on the fact that the stem (CString str) function, which would return the stemmed version of the CString, was needed in various locations without the need of the other porter functions.

The Parser class uses the StringTokenizer. StringTokenizer is an iterated-type class that iterates through the words and the punctuations. It distinguishes punctuations and words, and iterates through them separately. The Tag Class represents the tags. It has several utility methods that tell if the tag is of given type. Some examples are IsImage(), IsHeaderClose(), IsScriptClose() IsHeader(), which return true to the Parser class if the CString, received by the StringTokenizer class is was a HTML tag. TagTokenizer class is an iterator-type class to iterate through tokens in a string. HasMoreTokens and GetNextWord methods are used to iterate.

### 1.4.3 Support Classes

The Support Classes are all other classes in the IRV program that helps it gather the data and help process the data to make ready for output by the CDynLookView class more specifically its ::OnDraw(CDC \*pDC) member function.

Dialog Window Boxes:

1. CAboutDlg, is a class based on the IDD\_ABOUTBOX dialog window that

holds information such as version number, author of the program, etc. The way to access is through the IDR\_MAINFRAME Menu, which is the main menu of the program.

2. CHelp, is another dialog based class more specifically on IDD\_Help. This dialog box contains data about what the program does and how it works.
3. CKeyWords is how the user inputs the keywords the user wants search for in the search space the words are sent to variable storage space, and stemmed. Currently the software program takes up to three words as keys.
4. CStartURLDiag is the final dialog box implemented in the IRV. This dialog box reads in one CString and places it in the URL queue as starts the engine of the program. The engine takes the next (in this case only) entry in the URL queue and process it, and follows the 'general algorithm' explained earlier.

More MFC structure classes and general functions:

1. CDynLookApp is the programs main entry point; it in essence is the whole program. When the program is started the 'main' of the C++ program is inherited into this class and the constructor is what starts. The program then goes to our next class, CMainFrame.
2. CMainFrame handles all the windowing procedures and message mapping and sends the appropriate (internal to IRV) functions.

These conclude all the classes used in the IRV system. While all of them had more member functions, including private variables, each main point was described.

## FUTURE WORK

Information Retrieval Viewer is still in an early part of its life cycle. Initially setup for web mining type visualization the many ideas could be implement on a database system very easily. More approaches could also be implemented and used to determine how appropriate one versus another or combination to find the 'best' document or web page. For the programming itself we would like extended future implementation on an OpenGL system instead of a Microsoft Windows-MFC system to allow for a third dimension to graph other approaches such as similarity-based approach in addition to the key word/authority already done. Another note would be in the HTML Parser class to give perhaps a ½ point or so to our MatchWord count value if the word is in a larger font or a heading to symbolize that its not just another word mentioned in the document but that the word is more significant in the topic discussed.

## CONCLUSIONS

What our result showed was that the combination of our approaches, keyword, and authority had its place and worked well alone yet together gave the best document in the searched space. Keyword with stemming proved to give a much more accurate in the results. For Authorities, the longer the program ran the more correct authority value of the document became since it could only compute the value based on parsed documents. While other approaches were possible and tested to do well to show the users different ways to get to the best data this worked reasonably well especially considering the semantic space we were working and trying to show.



## .REFERENCES

- Borner, K. (2000). The Semantic Space of Online Course Descriptions. *Proceedings of ACM Digital Libraries*, 234-235. San Antonio, TX.
- Börner, K., Chen, Chaomei. (Eds.) (2002). Visual Interfaces to Digital Libraries [JCDL 2002 Workshop]. 2539 Springer, ISBN 3-540-00247-2, Top Ten Problems with Visual Interfaces to Digital Libraries, pp. 43-49. 16.
- Brian Amento, Loren Terveen, and Will Hill. (2000). Does "authority" mean quality: Predicting expert quality ratings of Web documents. *Proceedings of the Twenty-third Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 296-303. New York, NY: ACM Press.
- Chang H., Hsu C.C. and Hou C.L. (1998). Exploiting Hyperlinks for Automatic information Discovery on the WWW. *Proceedings of the Tenth IEEE International Conference on Tools with Artificial Intelligence*, 58-72. Taipei, Taiwan.
- Chen C., Czerwinski M. (2000). Empirical evaluation of information visualizations: An introduction. *International Journal of Human-Computer Studies*, 2000 Vol. 53, 631-635.
- Ding, C., He X., Husbands, Parry et al. (2002). "PageRank, HITS, and a Unified Framework for Link Analysis". *Proceedings of the Twenty-Fifth Annual*

*International ACM SIGIR Conference on Research and Development in Information Retrieval*, 353-354. New York, NY: ACM Press.

Dziadosz, S., and Chandrasekar, R. (2002). Do Thumbnail Previews Help Users Make Better Relevance Decisions about Web Search Results? *Proceedings of the Twenty-Fifth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 365-366. New York, NY: ACM Press.

Fowler, R. H., Tor, Y., Navarro, D., & Grabowski, L. (2003). Efficient text content extraction and browsing using the Abstract Text Viewer. *Proceedings of the International Conference on Internet Computing (IC'03)*, 633-639. CSREA Press.

Harman, D. (1987). "A Failure Analysis on the Limitations of Suffixing in an Online Environment" *Proceedings of 10<sup>th</sup> Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 102-108. New York, NY: ACM Press.

Kleinberg J. (1998). Authoritative Sources in a Hyper-Linked Environment. *Proceeding of 9th ACM-SIAM Symposium on Discrete Algorithms*, 110-116. Charlotte, NC: ACM Press.

Kraaji W. and Pohlmann R. (1996). "Viewing Stemming as Recall Enhancement", in *Proceedings of the Twenty-Fifth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 40-48. Zurich: ACM Press.

Maristella Agosti, Fabio Crestani, and Massimo Melucci. (1995). Design and

- implemenation of a tool for the automatic construction of hypertexts for information retrieval. Revised conference paper, *Dipartimento di Elettroica ed Informatica, Universit`a di Padova*. Italy, pp.178-184.
- Nation D., Plaisant C., Marchionini G. and Komlodi A. (1997). Visualizing Websites Using a Hierarchical Table of Contents Browser: WebTOC. *Proceedings of the 3rd Conference on Human Factors and the Web*, 201-211. Santa Monica, CA: Human Factors Society.
- Ng Y., Zheng A. X., and Jordan M. I. (2001). Stable algorithms for link analysis. *Proceedings of the Twenty-Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 388-394. New York: ACM Press.
- Porter, (1980). Algorithm for Suffix Stripping, *Computer Lab, 1987 Vol. 14*, 130-137.
- Shneiderman, B. (1997). Human Factors of Interactive Software. In *Designing the User Interface: Strategies for Effective Human- Computer Interaction*, Addison-Wesley, pp. 1-37. Holmquist, L. E., Fargrell, H., Busso, R. Navigating Cyberspace with CyberGeo Maps. *In Proceedings of IRIS 21*, Seby, Denmark.