CS 748 Implementation Report:

Design of a Multimedia Web Crawler

Submitted to: Prof. Charlie Clarke

Submitted by: Steven Engels
ID 94006290

Date: August 1, 2000

Table of Contents

Table of Figures

1 Introduction

As networks and Internet technology become more advanced, media files of various types are populating the Internet with increasing frequency. This is because the improved speed and capacity of both computers and network connections have made computer-based multimedia more accessible for many Internet users. Examples of this are the recent craze over MP3 sound files, as well as the increased ease with which video files are both produced and downloaded from the World Wide Web.

With this increase in media usage on the Internet, the demand for searching technologies that can locate these media files is also increasing. For example, search engine providers such as Lycos, Excite and HotBot have taken steps to provide multimedia-based search engines, while web sites such as Scour.com and ImageSurfer base their livelihood on multimedia searches.

In order to provide accurate and thorough indices of multimedia on the web, two main tasks need to be performed: exploring the Internet for multimedia files, and storing searchable databases of the file descriptions and locations. To date, most crawlers include multimedia searches as a component of their broader search engines. However, aside from isolated MP3 crawlers, the author and his colleagues have had difficulty in finding crawlers that search the Internet solely for multimedia files.

The purpose of this project is to design a crawler that would effectively perform a multimedia search of the Internet. If successful, the result will be a web crawler that not only finds and lists the location of multimedia files on the Web, but will also used this multimedia focus to perform the task more efficiently than a normal search.

2 Technical Details

This web crawler was designed on a UNIX platform using Java's Java Design Kit version 1.2.1.

3 Design

3.1 General Design

This crawler has three main objectives:

1. Perform a thorough search of the Internet, exploring as many links as possible, while avoiding pitfalls (e.g. bad links, cgi-bin redirections, etc.)
2. Find and record the location of any multimedia files found (images, sound, video), keeping in mind that some files are disguised as others to make detection by difficult for copyright reasons.
3. Use the discovery of multimedia files as a measure of the success of a search path.

The method for performing these three objectives is outlined in the following sections.

3.2 Searching the Web

The fundamental task when searching the Internet is to recursively explore all of the links found on previously searched Web pages, starting from an initial page or set of pages. To accomplish this, the author retrieved code for a basic web crawler from the Sun Java web site (http://developer.java.sun.com/developer/technicalArticles/ThirdParty/WebCrawler/WebCrawler.java). This code performed the following operation:

1. Construct the GUI (shown below)
2. Initialize Searched list, ToSearch list and Matched list
3. While the ToSearch list is not empty
   3.1 Remove the first ToSearch element, place it on the Searched list
   3.2 Check that the element is a valid HTTP URL
   3.3 Check that the URL is a readable text page
   3.4 Find all "<A HREF" links and check if they are valid
      3.4.1 If a link goes to a valid text/html file, record the link on the ToSearch list
      3.4.2 If a link matches the desired media file type, record it on the Matched list
4. If the number of Searched or Matched files exceeds the specified limit, exit

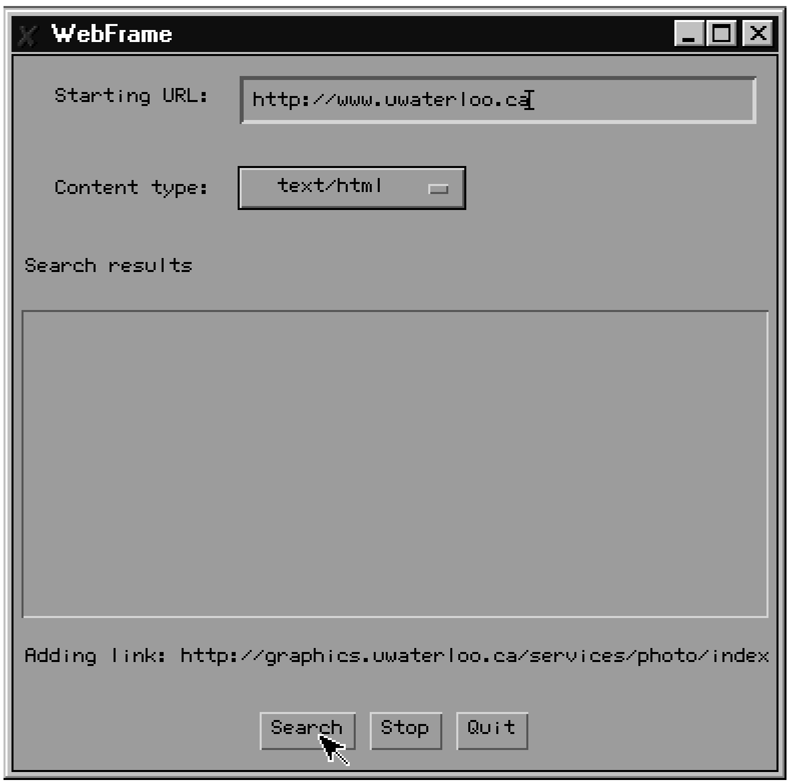

Figure 1: Screen Capture of Web Crawler GUI

In addition to checking if a page is a valid HTTP file, this implementation checks if the site is "robot safe", that is, if it allows crawlers and other Internet bots to search through its pages. A web site can specify this by including the file *robots.txt* in the home directory, which contains the word "Disallow" if crawlers are forbidden from entering.

Although the downloaded code provided the necessary structure, its functionality has several limitations:

1. It does not explore all of the tags that could lead to potentially interesting files
2. It exits at any unusual situation (i.e. badly specified URL, cgi-bin files, page disallowing crawlers)
3. It only searches for a few file types
4. The type checking method is unreliable, and misses many valid files
5. It only allows the user to start searching from a single Internet site

The first two limitations present serious reliability issues, since they make the crawler more prone to missing important files and failing prematurely. The third and fourth limitations result from the crawler not being focussed enough on finding multimedia files. The final limitation is less important, but still prevents the crawler from searching a wide variety of sites.

In order to overcome these limitations, the basic structure of the downloaded version was used, with modifications to most of the sections. The changes made are as follows:

3.2.1 GUI Changes

1. Starting URL modified to accept local file name, which contains the "seed" URLs to begin the search
2. Status messages modified to report a more complete description of the internal operations
3. "Quit" button added, so that GUI does not have to be terminated from command line

3.2.2 General Operation Changes

1. Several "break" commands removed to bypass non-threatening errors
2. Trace of crawler's operation recorded in history file (History.dat)
3. Output of crawler (searched, to search, matched lists) recorded in HTML output file (output.html)
4. Instead of simply storing URL names, the lists were changed to store lists of structures, where each structure stored the name of the URL, its children, and the URL's weight (described later).

3.2.3 Search/Discovery Changes

1. Files classified based on deciphering the file header information, not Java's `guessContentTypeFromStream` call
2. More files types searched (WAV, MIDI, AU, MP3, MPEG, Quicktime, AVI, GIF, JPEG)
3. Frames explored
4. Files embedded in page explored (e.g. image files, QuickTime files)
5. Search allows general media types to be found (e.g. all video, all audio, all images)

3.3 Search Enhancements

A reinforcement algorithm was added to the search mechanism described above, to determine whether it would enhance the results retrieved by the crawler. Each URL explored would be given an initial weight of one, which would increase if there was an increased likelihood of finding multimedia files at that URL. The lists of sites are then sorted so that sites with higher weights are explored first.

The weight of a web site could be increased in one of two ways:

1. The crawler finds a multimedia file of the specified type on the site (increases weight by one)
2. The crawler finds multimedia files of the specified type on related sites (increases weight by a fraction of the number of files found)

The first weight increase is straightforward. The weight of the site is increased by an amount proportional to the number of files found. Once this is complete, the second weight increase propagates a fraction of its weight increase back to parent and sibling sites. The principle behind this weight propagation is that if a current site links to another where multimedia files are found, it is likely that its other links would lead to multimedia sites as well. If one uses a tree structure to represent the links between sites, a weight increase at a leaf node could be propagated back to its ancestors and other related nodes, with the weight increase lessening as the distance between nodes increases.

The weighting algorithm not only updates the weights to a site's ancestors, but also to the children of these ancestors. This is based on the idea that if a site with many multimedia files has a given ancestor, the ancestor's descendants would be more likely to have multimedia files than the ancestor would, and the likeliness would increase as the distance from the ancestor approached that of the multimedia site. Simply stated, a site's cousins would be expected to have more similarities to each other than they would to their ancestors.

To implement this weight update, a weight factor is introduced. This weight factor is an integer that represents how many multimedia files one must find on a site before increasing the parents' and siblings' weights by one. This weight factor thus represents the number a site's weight increase is divided by when updating its parent's weight. Propagating weight increases back through the parents ensures that an ancestor's weight increase depends not only on the number of files found by its descendant but also its distance from the descendant.

The weight update algorithm has the following steps:

1. Calculate the weight increase for this site
2. If the weight increase is close to one, go to Step #5
3. Otherwise, find any parents to this site
4. For each parent, recursively perform Step #1 again with the current weight increase divided by the weight factor
5. Increase the current site's weight by the current weight increase
6. If the current site is at the same depth as the original site, exit
7. Otherwise, find all of the site's children
8. Recursively perform Step #5 on all of the children until reaching the original site's depth.

Steps 1-4 recursively find the ancestors of the current site, while steps 5-8 recursively update all of the children of the ancestors found. A list of updated sites is maintained to ensure that the weights of one site are not updated more than once.

An illustration of the result of updating the weights of a site is shown below.

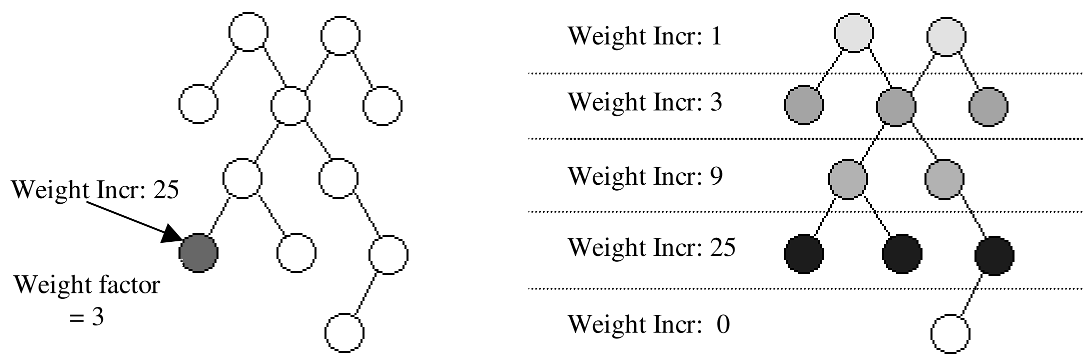

Figure 2: Illustration of Weighting Algorithm

4 Performance

Performance data for this crawler is difficult to obtain, since it is sometimes necessary to search through thousands of sites in order to acquire meaningful results. On top of this, certain test cases are best tested multiple times to eliminate external effects such as high traffic times or high server loads. Despite these factors, preliminary results are available, although there is still much more performance data that must be gathered.

In this case, performance is measured by the amount of time needed to find a specified number of multimedia files. Another possible measure is to specify a threshold number of sites for the crawler to search, and then count how many multimedia files are found by the time this threshold is reached. However, this threshold needs to be quite high to ensure that the crawler searches enough sites, which can be expensive in the time and computing resources required.

The most significant observation was that the reinforced version of the crawler performed worse than the regular version of the crawler in some cases. This was an unexpected result, but upon further inspection, these cases appear to involve sites with very large web page directories, where some of the pages contain the multimedia types being sought. These directory pages were a problem for the enhanced crawler, because every page in the directory that had multimedia files caused the crawler to update its weights for all the other pages. This did not change the way the directory was searched, since the result was that all the pages in the directory were reweighted equally. However, after multiple updates, the overhead cost incurred by the enhanced crawler meant that updating the weights would increase the time to search a directory, without improving the performance.

In cases with low branching factors, this effect does not occur, and the enhanced crawler performs better, as expected. It is hoped that, given a large enough search space, the effects of high branching sites would be absorbed in the larger network structure.

One other observation was that the enhanced version of the crawler would not perform as well in cases with a low limit on the number of matched files sought. This occurred because when the matched file limit was low, the crawler would halt before any effects of the enhancements could be seen.

5 Conclusion

This crawler design performs well at exploring the Internet and identifying the multimedia files it encounters. The reinforcement algorithm is helpful in focussing on areas that are more likely to contain multimedia files, although it is more effective as the search space of the crawler increases.

6 Future Work

The following are potential areas of expansion for this web crawler:

1. Use keyword matching (e.g. "music", "video", etc.) to enhance the weighting of a page
2. With the above enhancement, allow the crawler to learn what words are better indicators of multimedia files by observing which lead to these files more consistently
3. Use the size of discovered media files as a measure of how much to reweight a page
4. Allow the crawler to access information through forms and CGI scripts