Searching the World–Wide Web Using Signature Files

Justin Sun **James Mayfield**

Computer Science and Electrical Engineering Department
University of Maryland Baltimore County
Baltimore MD 21228–5398 USA
{sun,mayfield}@cs.umbc.edu

Abstract

A problem commonly faced by users of the World–Wide Web (WWW) is forgetting the path traversed to reach a previously read document. SWISS (**S**eeking **W**orld–Wide Web **I**nformation Using a **S**ignature File **S**earch) is a system designed to alleviate this 'lost document problem' by incrementally saving the contents of visited documents in a signature file index. SWISS allows the user to retrieve a previously read document via keyword search. Advantages of the system include parsimonious use of resources, capability for search on the entire text of indexed documents, and browser independence.

1 Introduction

The Internet and the World–Wide Web (WWW) have caused an explosion in the volume of readily accessible online textual information. The World–Wide Web is a globally distributed hypertext system. Originally intended as a communication tool for researchers, the Web has expanded into the fastest growing resource on the Internet in less than three years. According to statistics made available on the Web by the Graphics, Visualization, and Usability Laboratory at the Georgia Institute of Technology [8], as of January 1995, WWW data is the second most used resource on the Internet (just behind ftp).

In hypertext systems such as the WWW, browsing is an important method for locating documents [1]. The primary advantage of browsing is ease of use. Though browser programs have many features to offer, they lack the ability to perform content–based search across multiple hypertext documents. After reading numerous documents, a user might forget the title and location of a document, while recalling the essence of the text. When this happens, the user needs a tool that searches the *full text* of the document to find words contained in the document. This has become a pervasive problem for users who want to quickly locate previously–visited documents; we refer to this problem as the *lost document* problem.

Three techniques are currently available for addressing the lost document problem:

1. **Browsing:** In the browsing approach, the user attempts to recreate the path that led to the desired document by manually traversing links. While this approach can be successful, it is tedious, error–prone, and degrades rapidly as the length of the path to the desired document increases.
2. **General indexing:** In the general indexing approach, the user relies on an index built independently of the user's browsing. This technique works well for finding documents that were initially discovered by using the same index. Unfortunately, it is impractical to index the entire Web, or even major sections of it, because of the immense size of the information space. Current web–wide indexes restrict their coverage to a subset of the available documents. This restriction is performed either manually (as with Yahoo [4]), or programmatically (as with Lycos [7]).
3. **Hotlists and bookmarks:** Hotlists and bookmarks provide the user with a way to explicitly save references to particular documents. Unfortunately, hotlists and bookmarks supported by Web browsers do not provide an adequate solution to the lost document problem. These mechanisms are useful only if users consciously save the desired links; in real life, the links users remember to add to a hotlist are but a subset of the links they later want to retrieve. Furthermore, hotlists and bookmarks typically save only the title of each document; the body of a document is searchable only by first retrieving the document.

Thus, none of these three standard techniques works well for the lost document problem. A solution to this problem must exhibit the following characteristics:

- **Whole–document indexing:** Because document titles comprise just a few words, and because they are usually not designed with indexing in mind, titles are insufficient as index entries for solving the lost document problem. The full text of a document must be searchable for users to have a reasonable chance of finding a document when it is lost.
- **Personalization:** The lost document problem is essentially a personal problem, because

a document can only be lost by a user who has previously found the document him/herself. Thus a solution to the problem must be tailored to individual users.[1]

- **Incremental update:** A program that helps users recall previously visited hypertext documents needs to update its index continually and incrementally. The typical user tends to read web documents in a limited number of short sessions. Therefore it is important to update the user's index as new documents are visited (*e.g.* after finishing a session). Doing so not only keeps the index up–to–date, but it also allows a single document retrieval to be used for both display and indexing.
- **Space efficiency:** a program that solves the lost document problem must be able to index the full text of all visited documents in a small space. Although mass storage devices have become cheaper in the last few years, it is not reasonable to expect all users to have access to large storage devices. Furthermore, because each user requires his or her own index (see *Personalization* above), there are no economies of scale to help limit space utilization (as there are when many users access the same general index).
- **Bandwidth efficiency:** It is impractical to download a Web document each time it is to be searched. A solution to the lost document problem must be parsimonious in its use of network bandwidth.
- **Unobtrusiveness:** Finally, such a system must build the index in an automatic and non-intrusive way. It should retrieve *any* previously read hypertext document without requiring the user to explicitly save the link.

The SWISS System (**S**eeking **W**orld–Wide Web **I**nformation Using a **S**ignature File **S**earch) is a document indexing system that conforms to each of these desiderata. In the next section, we discuss the main full–text indexing methods in current use. Section 3 describes our application of one of these methods, signature files, to the lost file problem.

2 Full Text Search Methods

Many options for indexing text documents exist. The three most popular methods are linear search, inverted files, and signature files.

2.1 Linear search

Linear search, also called the brute force method, requires no index file. It is the slowest of the three methods because each word of the original file must be compared against the query words. Advanced pattern matching software and faster processors have made the linear search reasonably fast for an individual's personal files [6]. It is not viable as a solution to the lost document problem though, because it would require significant network bandwidth for each query.

2.2 Inverted files

Inverted files [9] are by far the most widely–used indexing structure for information retrieval. After common English words (such as the articles 'a', 'an', 'the') have been removed from the set of words found in the corpus, an inverted file is constructed by ordering the remaining words alphabetically, and associating with each word a set of references to instances of the word in the corpus. Inverted files have the advantage of rapid retrieval, but require about 50–300% storage overhead [3]. In addition to these space requirements, inverted files require substantial processing time for index maintenance (thereby violating the incremental update requirement).

2.3 Signature files

Signature files contain collections of bit strings which represent the contents of a document or part of a document. A *signature* is a fixed–length bit vector consisting of ones and zeroes. Word signatures are bit vectors that correspond to a given word. The *SWISS* system uses the simplest type of signature file, the sequential signature file (SSF). Constructing an SSF requires the following steps:

- The text to be indexed is split into distinct words. Words that appear on a stoplist, which consists of the most common English words, are removed.
- An n bit signature is generated for each remaining distinct word. Initially, all n bits are set to zero. Then, k positions are set to one using k different hash functions, where $k < n$.
- All of the word signatures are combined through a process called *superimposed coding* to produce a document signature. Superimposed coding is done by performing the logical OR operation on all word signatures to create a document signature.
- The newly–created document signature is appended to the signature file.

Signature files possess several advantages over other indexing methods: they require little storage overhead, they can easily accommodate insertions and deletions, and they provide relatively fast retrieval times.

To locate documents containing particular words using signature file search, the user first enters a set of query words. Each query word generates a word signature using the same hash functions that generated the document signature. These word signatures are then OR'ed together to create a query signature. To determine whether a document contains

[1]While it can be useful for a user to search for documents that have been perused by others (especially if the others have interests similar to that user), doing so is not a solution to the lost document problem.

the query words, each of the document signatures is compared with the query signature. If every position that contains a one in the query signature also contains a one in the document signature, then it is likely that the document contains the query words. Documents with signatures that meet this criterion are flagged for retrieval.

Although signature file search will never miss a document containing the query words, it is possible that it will inadvertantly retrieve documents that do not actually contain the requested words, but whose signatures happen to contain the desired pattern of ones. The number of such false positives can be reduced by increasing the signature size, at the expense of increasing the size of the index.

3 The SWISS System

The purpose of the SWISS system is to help the user recall previously visited hypertext documents. The system is particularly valuable if a user recalls reading about a topic of interest, but forgets the links traversed to get there. SWISS uses signature file search to retrieve documents of potential relevance to a user's query.

The current implementation of the SWISS system comprises the following components:

1. An email filter that accepts and processes user indexing requests. This program reads the list of documents that the user has read and spawns a process that downloads the appropriate Hypertext Markup Language (HTML) source files over the network for indexing. This approach is used to make indexing browser–independent. As browsers that allow proxies and other filtering methods become generally available, we plan to integrate indexing more tightly with browsing, while maintaining browser independence.

2. A program which accepts a single HTML document as input, builds the appropriate document signature, and updates the signature file index with this new signature. This program may periodically update the signature file to reflect modifications and deletions of remote HTML source files.

   After retrieving the HTML source, the first step in the indexing process is to remove the markup tags and break the document into individual words. Then, common English words such as articles, prepositions, and verbs that appear on a *stoplist* [5] are thrown out. It is senseless to store any words appearing on the stoplist; such words occur in nearly every document and therefore cannot distinguish one document from another.

   The SWISS system then creates fixed length signatures. Each word is encoded by a set of k distinct hash functions, each of which sets a one bit in the signature vector. For efficiency, Faloutsos recommends using a variation of the document signature called a *block signature* [2]. A block signature indexes a portion of a text file (*e.g.* five word signatures logically OR'ed). Although the block signatures are of fixed length, the number of block signatures is directly proportional to the text size. The document signature in this case consists of the set of its block signatures. The SWISS system uses the block signature method to compensate for the wide variance in the size of HTML documents.

   SWISS builds a full text index of each HTML document minus the markup tags. Once the document signature has been generated, the original hypertext source is discarded.

3. An HTTP server that accepts queries and performs a signature file search of the index. The server returns a list of anchors that are likely to contain the query words.

   On a workstation that supports a forms interface, the user can specify query words on the SWISS query form. After submitting a request, the system returns a list of documents that match the query signature. The user may then decide based on document title which documents are most likely to be relevant to the search. Thus, a final highly–constrained browsing phase might be needed if more than one document matches the user's query.

Together, these components provide a system that meets the requirements enumerated in the Introduction to solve the lost document problem.

4 Conclusion

SWISS is a browser–independent information retrieval tool that helps users find prevously–perused web pages. SWISS offers several advantages to its users. First, it helps users find information that they may have remembered reading, but whose location they have forgotten. Second, the indexing mechanism is non–invasive, occurs asynchronously, and may be incrementally updated after the user finishes browsing. Third, the index is maintained automatically, without explicit user intervention. Finally, searching for documents is quick, and the index file occupies little space compared to the documents indexed.

We are currently pursuing the use of a new type of signature file, called an *n–gram signature file*, with the SWISS system. N–gram signature files use n–grams (sequences of n adjacent characters) as terms instead of words or word stems. We anticipate that the new system will be more accurate and more robust to typographical errors, with only minor increase in the size of the index files.

References

[1] Tim Berners-Lee, Jean-Francois Groff, and Bernd Pollermann. World–Wide Web: The in-

formation universe. *Electronic Networking: Research, Applications and Policy*, 1(2), Spring 1992.

[2] Christos Faloutsos. Signature files: An integrated access method for text and attributes, suitable for optical disk storage. Technical Report CS–TR–1867, University of Maryland, College Park, June 1987.

[3] Christos Faloutsos and Raphael Chan. Fast text access methods for optical disks: Designs and performance comparison. Technical Report CS–TR–1958, University of Maryland, College Park, December 1987.

[4] David Filo and Jerry Yang. Yahoo. `http://www.yahoo.com/`.

[5] Christopher Fox. Lexical analysis and stoplists. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval Data Structures & Algorithms*, chapter 7, pages 102–130. Prentice Hall, 1992.

[6] Udi Manber and Sun Wu. Glimpse: A tool to search through entire file systems. Technical Report TR 93–34, University of Arizona, October 1993.

[7] Michael L. Mauldin. Lycos: The catalog of the internet. `http://lycos.cs.cmu.edu/`.

[8] James Pitkow. GVU Center NFSNET statistics. `http://www.gatech.edu/gvu/stats/NSF/merit.html`.

[9] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison–Wesley, 1989.