From Hypertext to Flat Text: A Tool for Document Construction

Jian Yang, Dept of Computer Science, University College, UNSW, Australian Defence Force Academy, Canberra ACT 2600, Email: jian@csadfa.cs.adfa.oz.au

Wanli Ma, Computer Sciences Laboratory, RSISE, The Australian National University, Canberra, ACT 0200, Email: ma@cslab.anu.edu.au, Home Page

Richard P. Brent,Computer Sciences Laboratory, RSISE, The Australian National University, Canberra, ACT 0200, Email: rpb@cslab.anu.edu.au, Home Page

Keywords: WorldWideWeb, Hypertext, Internet, Hypertext Printing, Document Construction

1 Introduction and Motivations

Hypertext [1,HREF1], since it was forged in 1965 by Ted Nelson, has become a hot research topic, especially when it was recently combined with the *World Wide Web* (WWW) [HREF2]. People enjoy navigating the WWW ocean for varieties of information by just clicking their mouse on computer screens. The hierarchically linked pieces of information allow users to browse through a network of chunks of information. The information is provided both by what is stored in each web site and in the way that information sites are linked to each other. A hypertext file format, written in HTML (Hypertext Markup Language), consists of links among documents to support the hypertext data organization. Those documents are identified over network by using URLs (Universal Resource Locator).

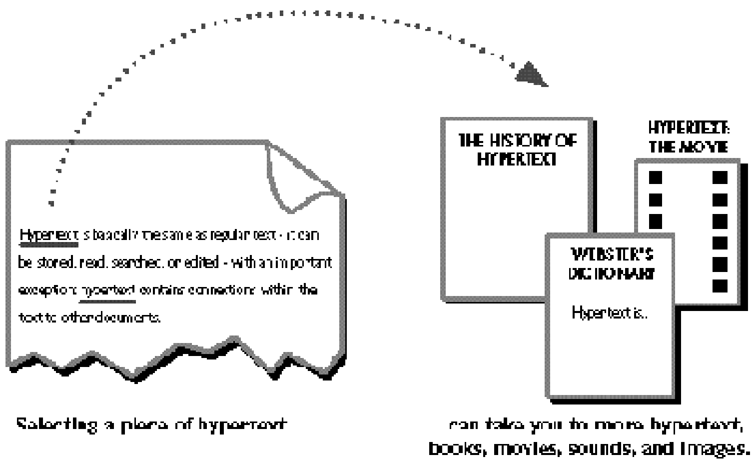

Figure 1: A Hyperlinked Document

Hypertext provides a way to transcend the limitations of linear structured documents by giving users the freedom to have their own chosen sequence of browsing a hyperlinked document (See Figure 1 [2]). In other words, the users do not have to be restricted to the *head-to-tail* way of reading as in a linear structure. Powerful as the hypertext structure is, a linear structure is still needed for hard-copy printing. When a user finds an interesting document, say, a book, in the Internet and would like to have a hard-copy of this book, its hierarchical structure makes printing a tedious task: the user have to manually follow each hyperlink, e.g., each chapter (then perhaps each section within a chapter), in the main page of the document, print each piece of the document separately, and finally, assemble them together to get the whole book. Even worse, the assembling-together might not be possible without physically cutting and pasting if a chapter is linked from the middle of the main text or a section from the middle of a chapter. Therefore a tool which can automatically follow the hyperlinks in a main document and replace each link with the actual sub-document in the final output is desirable.

In this paper, we promote such a tool called *H2FDoc*. The bulk of this paper is devoted to presenting the tool and its capabilities. The automatic construction of the final flat output, also in HTML format, is based on a set of heuristic rules which specify the depth of link searching, the means of dealing with cyclic links and images etc. We believe that this kind of tool for document construction is necessary as a supplement to WWW browsing and authoring tools.

The rest of the paper is structured as follows: Section 2 presents an outline of H2FDoc, and Section 3 discusses the searching algorithms and heuristic rules used to build H2FDoc. Section 4 gives two testing results of H2FDoc on the examples of Netscape handbook and the Third International WWW Conference Proceedings. Finally, Section 5 summarizes the paper with a conclusion and our future plans.

2 The Architecture of H2FDoc

H2FDoc works in a similar way to a web browser. However if there are hyperlinks embedded, H2FDoc replaces them with the actual sub-documents and *recursively* browses their offspring links according to the heuristics rules until a flat document is achieved. The architecture of H2FDoc is shown in Figure 2.

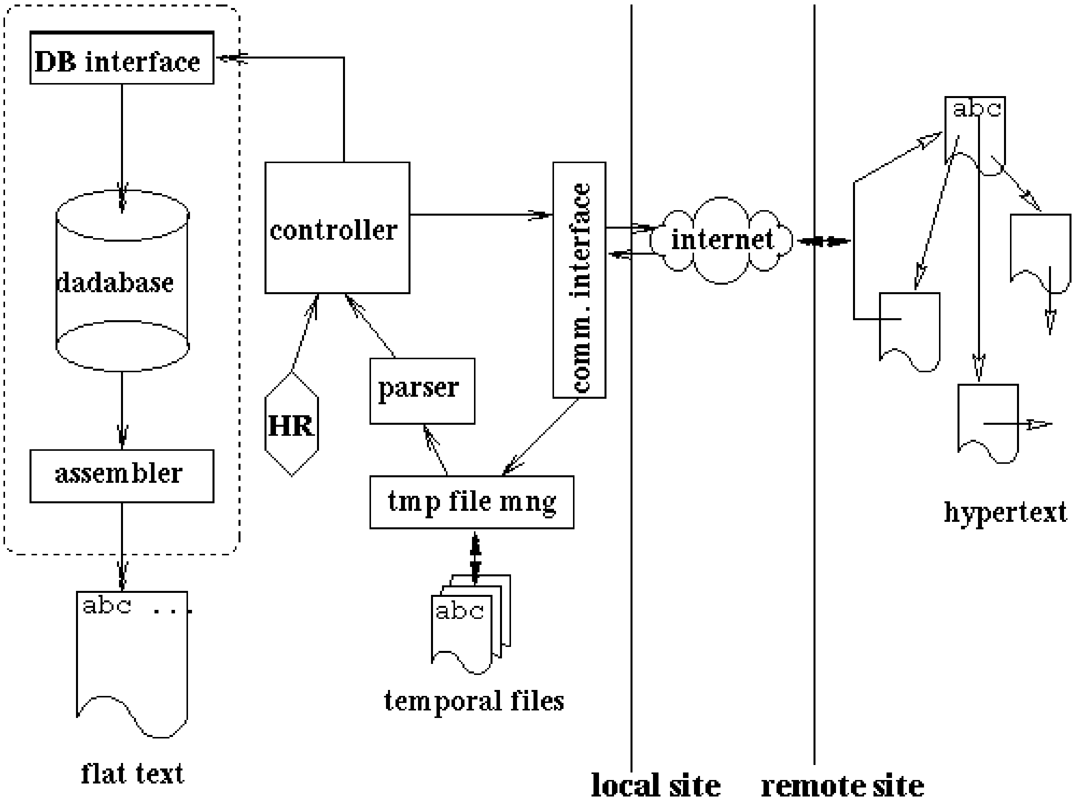

Figure 2 The Architecture of H2FDoc

The main components of H2FDoc are in the following:

- **Communication interface**: it is responsible for making connections to, sending requests to, and receiving data packets from a remote host, where the interesting document resides. The interface uses the socket library of a Unix operation system to communicate with the Internet. The data received are passed to the temporary file manager.

- **Temporary file manager**: it maintains a number of temporary files which correspond to the files from a remote host. We need to store these temporary files as HTTP is a non-stop protocol. It transmits a data packet without any stop or interruption. The manager takes the data packets (HTML files) from the communication interface and sends them to different temporary files, which will be removed after the current execution. The manager is also responsible for the switching and recovering, on behalf of the Controller, among those temporary files.

- **Parser**: it analyses the lexical and syntactic structures of an HTML file from one of the temporary files to determine whether the currently processed HTML tag is of interest or not (see next section for explanation of interesting and uninteresting tags). The Parser sends data tuples, which contain current text, the tag name, the tag type, and URL etc. information (details will be discussed afterwards), to the Controller. For any interesting tuple (tag), the Controller will react accordingly; otherwise, it just simply passes the tuple to the next stage. This part is developed with the help of *lex* and *yacc* utilities.

- **Controller**: it takes actions on the interesting tuples (tags) based on a set of heuristic rules (see next section) and passes the uninteresting ones. For example, when an interesting tag, say, `<a href="URL">`, comes, the Controller will ask the communication interface to contact the remote server specified by the `"URL"` and get the corresponding file to a temporary file, and then ask the temporary file manager to preserve the environment of the file currently being processed and switch the "current file pointer" to the new file. Processing of the old file will not be resumed until processing the current file is finished.

- **DB interface**, **database** and **assembler** are intended to systematically preserve the information extracted from an HTML file, but they have not been developed yet in the current prototype.

Apart from the above components, a data tuple is used to record the following information:

from	line	col	font	face	type	spelling	comment

where

- `from` records the URL address of the original file;
- `line` and `col` are the line and column numbers of the processed tag in the original file;
- `font` is the current font of the string;
- `face` is the current face of the string;
- `type` gives the link type of the tag. For example, the type could be a plain text (`PT`), a hypertext link (`HL`), and a image link (`IL`) etc;
- `spelling` is the real spelling of the tag or a plain text;
- `comment` can be used to provided some extra information if necessary.

When these tuples become records stored in a database, they can be used to control the process of the Controller, be queried by users, and control the output presentation of a flat document such as page numbering and indexing. However this facility has not been implemented in the current prototype yet.

3 The Search Algorithm

The main component of H2FDoc is the Controller, which determines the presentation of the output document. To achieve this, it has to be able to differentiate between those tags in HTML documents which will be ignored (*uninteresting tags*) and those which will be further processed under the heuristic rules (*interesting tags*). In the following sub-sections, we discuss the classification of tags and the heuristic rules for hyperlink searching.

3.1 A classification of HTML tags

In order to replace the links with the actual documents they point to, we have to analyze all the tags in a source document file to decide whether they should be processed or not. Here we classify tags as:

- **Uninteresting tags** are those general tags which specify the structure of a document (e.g., `<h1>`, `<br>` and `<p>` etc.), presentation format (`<em>`, `<b>`, and so on), background and colors, special characters, tables, and forms. We simply pass uninteresting tags to the next stage without taking any action.

- **Interesting tags** are the pointers or marks of the linking documents as follows:
    1. `<a href="URL">XXX</a>`: in this case, the `XXX` should be replaced by the content this `URL` is pointing to: when the tag is met, the Controller will interrupt the processing of the current document and get this new file, pointed by this `URL`, and then work on it instead. The interruption point will be recovered after the processing of this new file is finished.
    2. `<a href="URL#abc">XXX</a>`: if it points to a particular portion of a different document which has not yet been received by the temporary file manager, the same action will be taken as above; otherwise, the tag is regarded as an uninteresting one.
    3. `<a href="#abc">XXX</a>`: it points to a different portion of the same document, so no action will be taken. The tag can be considered as an uninteresting one, but it can be useful if an index should be created. We will address the question in a future paper.
    4. `<img src="URL">`: it points to an image. We need to replace it with the image, i.e., get the real data of the image.

When H2FDoc goes down to follow an interesting tag, say, `<a href="URL">XXX</a>`, it may find out new interesting tags and has to go further down, or *recursively*, to follow those new tags. A document most likely contains links pointing to its ancestor documents (in the sense of hierarchical structure of hypertext), or there might be pointer loops in the documents. H2FDoc should be able to recursively follow the interesting tags without being trapped in the loops. In the following section, we discuss the rules for hyperlink searching.

3.2 The heuristic rules for recursively link searching

If there is no restriction imposed, the hyperlink searching may go on forever. Therefore we need rules to ensure termination. In the following, we list some rules which have been implemented in H2FDoc:

- level restriction by setting `LEVEL` threshold: in general, most publications tend to have less than 5 levels (e.g., chapter, section, subsection, etc.). Therefore, the searching is restricted to the depth less than this threshold. Take the examples of Netscape handbook [HREF3, HREF4] and the 3rd international WWW conference proceedings (3WWW) [HREF5], both are of two level hypertext structure although they use a different directory structure. `LEVEL` can be set by a user before H2FDoc starts.

- no remote (or outside) site search: we believe that a document together with its **directly related** parts, such as sections and subsections, normally is kept in a same site; therefore links pointing to a remote (or an outside site) document, e.g., author links in 3WWW, should be ignored.

- no recursive search on a clickable picture: if the link points to an image, H2FDoc simply gets the picture without going any further, because most clickable pictures in a document just let you do zooming or getting parts of the picture.

- no repeating search: if a link points back to a document which has already been processed, this link is ignored, i.e., no further search is made.

- no back search: H2FDoc does not follow any link pointing back to the ancestor documents of the current one to avoid search loops; furthermore, H2FDoc will never search any documents which are beyond the offsprings and the siblings of the main document, as we think that the main document should be the root of the whole files.

4 Examples

In this section, we give two examples--**Netscape Handbook** and **3rd WWW Conference Proceedings**--and discuss our experiences and the capabilities of H2FDoc.

4.1 The Netscape Handbook

There are three sections in the Netscape Handbook: Tutorial, Reference and Index. It can be easily checked by clicking the "`Handbook`" button in a Netscape browser. The logic structure of the handbook is in Figure 3, where the arrows are representing the hyperlinks in the original text.

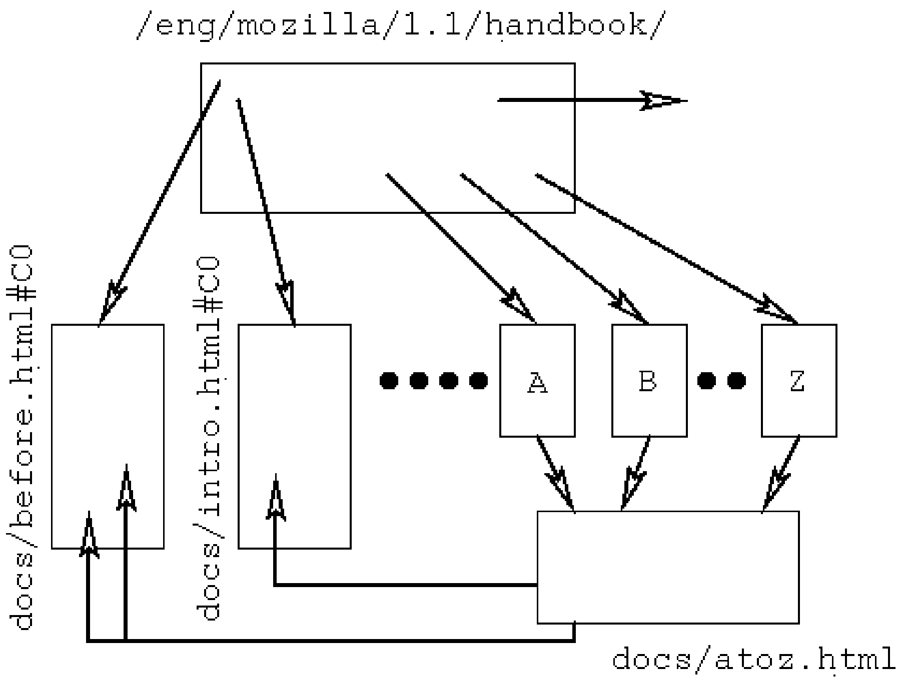

Figure 3: The Hyperlinked Structure of the Netscape Handbook

From the structure we can see that this is a network-linked text document. In this example, the level threshold `LEVEL` is set to 2. After the execution of H2FDoc, this document is flattened, as shown in Figure 4.

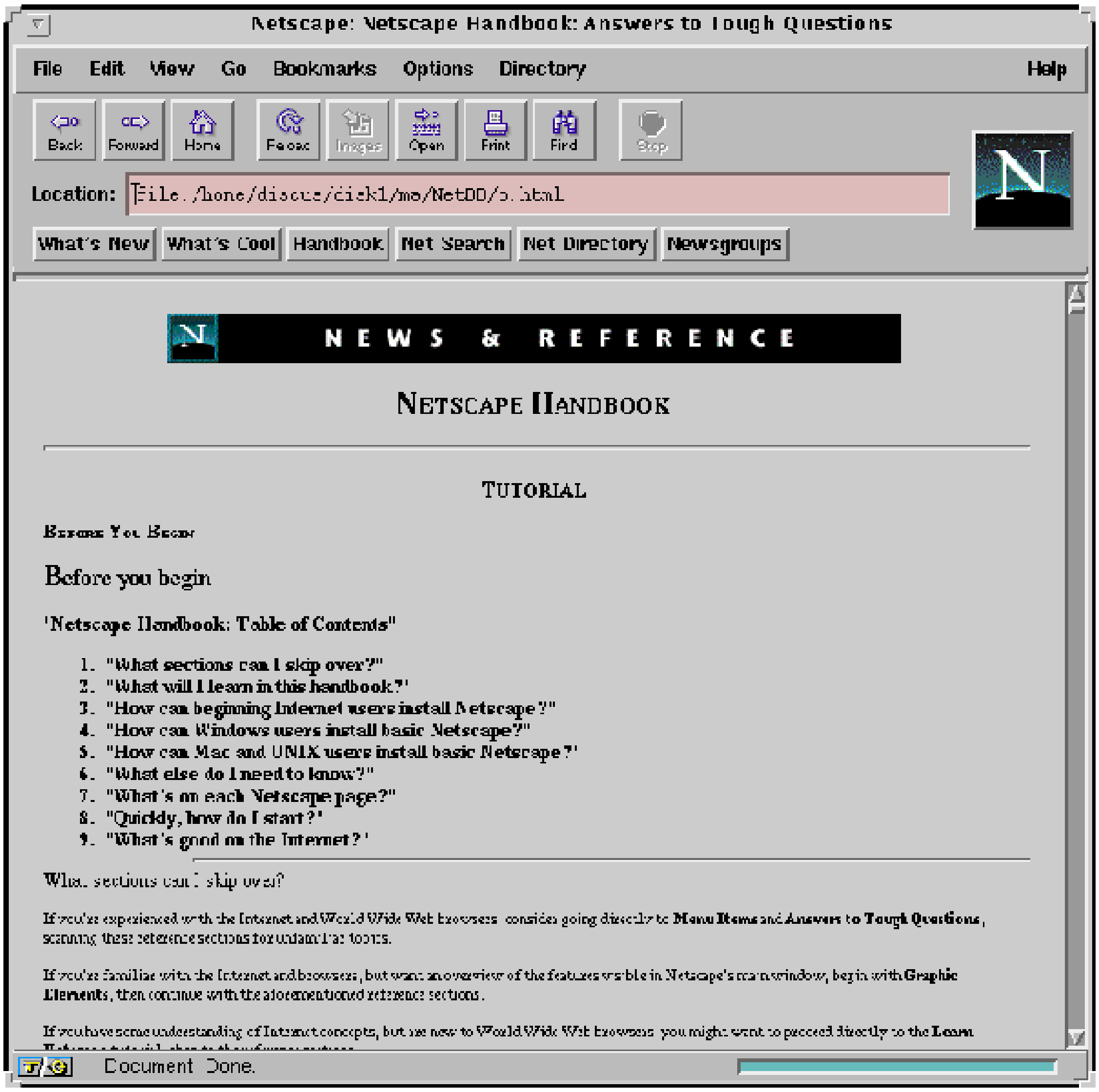

Figure 4: The Flattened Netscape Handbook

The image band "**News & Reference**" still can be clicked, and will take the user to the homepage of Netscape. The only problem with this example is that the Index section is lost. This is because with the current heuristic rules, if a tag points to document which has been traced before, it is ignored. Future development of H2FDoc should be able to pick up the index pointers and make the output similar to normal publications, such as a book.

4.2 The 3rd WWW proceedings

We have also gone through the third International WWW Conference proceedings via its abstract list:

`http://www.igd.fhg.de/www/www95/proceedings/papers/abstract.html.`

By replacing the hypertext links in the file with actual files, we can eventually get the whole proceedings. The heuristic rules work well on it except that each abstract is duplicated: one after the title link in the main text and the other in the actual content to which the title is linked. Although one of them could be removed by a smarter H2FDoc, the current prototype cannot handle this problem.

5 Conclusion and Future Work

In this paper, we present a tool, H2FDoc, which can be used to flatten a hypertext document so that the whole document can easily be printed.

We believe such tool is necessary for document construction in the environment such as WWW. Authoring and electronic publishing tools using hypertext techniques are becoming popular. Unfortunately, the reverse need for converting a hypertext to a flat-text has not yet attracted much attention. When a user wants to download and/or print a hypertext document, H2FDoc is very helpful.

The implementation of H2FDoc is only at its initial stage. Some functions such as error handling, duplication removing, heading and font adjustment will be developed in the future. Among them, the error handling feature is the most desirable one. Because HTML documents are written by different authors with different background, errors are inevitable. Take the following specification as an example (from a handbook about WWW),

```
<a href=fig6-2.gif"><IMG SRC= "fig6-2.thumb.gif"></A><BR>,
```

A double quote mark after the `href=` is missing. Netscape works well on it, but H2FDoc can not correctly interpret the URL of `"fig6-2.gif"`, and produces strange output.

H2FDoc currently assumes that all the related parts of a document are at the same site, but this is not always the case in the real world. For example, a company or a government might run a number of WWW sites. The assumption is the essential pre-condition to run the heuristic rules of this paper; otherwise, automatic browsing may never stop. The current heuristic rules are by no means perfect. A future version of H2FDoc will introduce the mechanism of *annotation*, by which we allow users to provide some useful information in addition to those heuristic rules to guide the recursively browsing. With those annotations, H2FDoc can make better decisions such as that it needs to go to some outside sites while some inside-pointing links are not necessary to browse. The annotations can also be used to provide the information on how to prepare the final output presentation.

Acknowledgements

The authors thank the anonymous referees for their illuminating comments on the first version of this paper.

References

[1] N. Woodhead. Hypertext & Hypermedia: Theory and Applications. Addison-Wesley, 1991.

[2] K. Hughes. Entering the World-Wide Web: A Guide to Cyberspace. Honolulu Community College, Oct., 1993.

Hypertext References

HREF1
: http://www.isg.sfu.ca/~duchier/misc/hypertext_review/index.html -- V. Balasubramanian. State of the Art Review on Hypermedia Issues and Applications.

HREF2
: http://www.leeds.ac.uk/ucs/WWW/handbook/handbook.html -- B. Kelly. Running a WWW Service.

HREF3
: http://home.netscape.com/index.html -- Netscape Homepage

HREF4
: http://home.netscape.com/eng/mozilla/1.1/handbook/ -- Netscape Handbook

HREF5
: http://www.igd.fhg.de/www/www95/proceedings/proceedings.html -- The Proceedings of the Third International WWW Conference.

Copyright
