The Journal of Natural Science,
Pai Chai University, Korea
Volume 8: 101 ~ 107, 1995

Conversion of linear, paper-based documents into Hypertext

Jin-Soo Kim, Dong-Won Park
Depatment of Computer Science, PaiChai University

선형문서를 하이퍼텍스트문서로 자동변환시키기 위한 연구 및 구현

김진수, 박동원
배재대학교 전자계산학과

The purpose of this work is to develop automatic techniques for converting linear, paper-based documents to a non-linear format suitable for use in hypertext systems. The selected document was paritally converted to hypertext manually, and a prototype was created using the rules derived from the manual conversion process. The full conversion was divided into three passes: correcting the electronic linear form of the document, generating a listing of the links in the document, and creating the hypertext document. Passes 2 and 3 were entirely automatic. From this study, it may be concluded that many classes of paper-based documents can be automatically converted to hypertext.

정보의 양이 늘어남에 따라, 필요한 정보를 빠르고 쉽게 추출할 수 있는 하이퍼텍스트 문서화에 대한 요구는 증가하고 있다. 기존의 서류문서를 하이퍼텍스트 전자문서로 변환시키기 위한 기술을 고찰하고 이를 구현하기 위한 소프트웨어를 개발하였다. 이 변환 작업은 세 과정으로 나누어져 있다. 첫 번째 과정에서는 스캐너를 이용하여 서류문서를 일단 전자문서화 시키고, 두 번째 과정에서는 첫 번째 과정의 output인 선형 전자문서를 전역 하이퍼텍스트 전자문서로 변환시킨다. 세 번째 과정에서는 이를 타겟 하이퍼텍스트 시스템의 포맷에 맞게끔 다시 변환시킨다. 이 작업을 통하여 실현가능성을 증명하였고, 또한 일반화 시키기 위한 문제점들을 제시하였다.

Key words : hypertext, hypermedia, hyperties encyclopedia

I. Introduction

Hypertext is a method of representing non-linear text using windows and machine-supported links between the windows. It is gaining interest as a tool for effectively accessing the large amount of information that is available today.

One of the problems with the initial creation of a large hypertext database will be converting existing documentation, which is typically linear, into a hypertext form. This paper describes the automatic conversion of an existing, paper-based document, the AIX user's manual, into a hypertext form.

The project is divided into three phases: the initial project definition, the prototype, and the actual conversion. Again, the conversion is divided into three "passes" where each pass processes the electronic form of the AIX manual. The first pass corrects errors in the document. The second pass produces information for use by the third pass which produces the hypertext database.

Hypermedia is an extension of hypertext which includes various forms of media such as graphics, animation, video and sound. Hypertext is not a new idea, but it has only been recently that technology has begun to approach the point at which computers can effectively support what humans have always done manually-creating and following links between pieces of information.

As an illustration, imagine yourself at a computer console reading an article which contains several references to figures, articles, and books. Now suppose that you could annotate the information and create your own links.

There are a few terms with which the reader should be familiar:

1.1. Nodes

windows on the screen are associated with nodes(objects) in a database. Each node typically holds a single idea or piece of information although one node could contain as much as an entire document or as little as a single word.

1.2. Links

Links form the interconnections between nodes. Using the interconnections, nodes can be organized in two basic ways: hierarchical(as in an outline) or non-hierarchical(as in a graph). Preferably, a combination of these two basic link types is available on a particular systems. Links may be bi-directional or uni-directional with a source and a destination(sink). A node may have multiple source links embedded in its text and may have several destination links pointing to it. If links are bi-directional, one can find all references to a node simply by inspecting the incoming links. Destination links may point to a location or region within a node(into-links) or only to another node(to-links). The user should be able to follow links easily(by "mousing" on them, for example) and the destination node should become active. Another aspect if links is that they may have types such as comment-link or reference link. This typing ability can facilitate searching and retrieval although it is the user's responsibility to use typed links appropriately.

II. Defining the Project

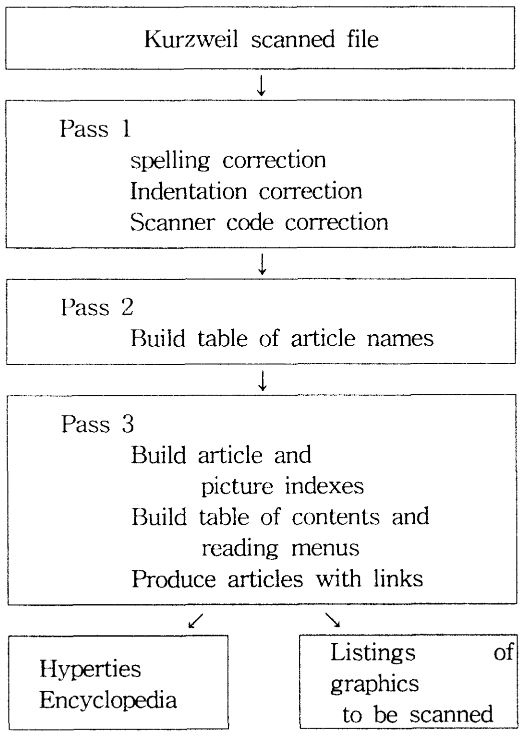
Most written information is in the form of Linear Paper-Based Documents(LPBD's). The cost of producing, distributing, and maintaining LPBD's is becoming prohibitive. Releasing updates is a cumbersome and time-consuming process. Many feel that hypertext is a more natural way of organizing information than LPBD's. As hypertext becomes more widely used, authors will write non-linearly rather than linearly. To speed the creation of large hypertext databases, it is necessary to convert existing documents into hypertext form. This may be done by someone rewriting the document or through an automatic process. The AIX project is part of a research effort to show that, at least for some classes of LPBD's, it is possible to automatically convert a linear document into non-linear form.
The implementation included all programming tasks as well as analyzing the target storage system and correcting the electronic linear form of the selected paper-based document. We developed the algorithm used to convert document to the intermediate format. Then we used this format to create nodes and links imported into hypertext system available such as Guide or Hyperties.

Since we assumed that the LPBD would not already exist in an electronic form, part of the project involved converting the text to ASCII using an OCR scanner and digitizing the figures using a graphics scanner.

Ⅲ. The Automatic Conversion Process

1. Planning the Automatic Conversion

Using the information gained from prototyping, we decided that three "passes" of the scanned text would be needed for the conversion process. Pass 1 involved correcting the scan file while Passes 2 and 3 were the actual conversion programs.


Kurzweil scanned file
Pass 1
spelling correction
Indentation correction
Scanner code correction
Pass 2
Build table of article names
Pass 3
Build article and picture indexes
Build table of contents and reading menus
Produce articles with links
Hyperties Encyclopedia
Listings of graphics to be scanned

2. Pass 1

Originally, Pass 1 was going to be a very simple process of replacing non-ASCII characters in the scan file with ASCII characters. For Example, the Kurzweil stored bullets("•") as "$.$"., which was easily changed to a dash with a processor. Also, most errors caused by the Kurzweil misleading characters could be caught and corrected with a spell-corrector.

After the prototype, We decided that we needed to be able to use the end-of-paragraph markers and that the indentation needed to be consistent from page to page. This meant going through the scan file line by line and correcting these types of errors. In addition to end-of-paragraph markers being placed within paragraphs, we found that they were often missing from lines within sections of single-spaced program code.

Because we had not defined a set of rules for the Kurzweil operator, there were inconsistencies in the marking of figures, boxes, and arrows.
So, as we were checking for errors; we inserted "$$$FIGURE$$$" wherever figures needed to be scanned and removed the lines of dashes, asterisks, and other characters which had sometimes been used to mark boxes and arrows.

For the prototype, we had used a character counter to assure that the articles would not exceed the Hyperties size limit. Since this did not decide the articles intelligently, we inserted markers, "$$$FORCE$$$", which would signal the conversion program to force a new article before the end of the reading section in cases where the article would have exceeded the size limit.

3. Analyzing the Hyperties Storage System

As we have mentioned, one of the main reasons for choosing Hyperties was its simple storage system. For this part of the project, we used the Author to create several articles with different characteristics such as no synonyms and multiple synonyms. We then used PCTools to print(in hexadecimal and ASCII) all the files that the Author had created and examined each form to determine its format. The conclusions we have reached are based on examples and may not cover all situations. The rest of this section summarizes the contents of each type of file.

3.1. Picture and Article Files

There are three main types of files in the Hyperties system: picture file(*.PIC), article files(*.FIL), and system files(*.TIE). Picture files contain the bitmap image of the figure. Each article file contains the heading, definition, and text for an article followed by a list of article file names, author notes, and list of picture file names. We used the article name(the heading of the first chapter / section / subsection in the article) and reading title for the heading but left the definition part blank, since useful definetions of articles could not be automatically created. Unfortunately, the Browser does not check to see if the definition is empty, and the user must still type the return key twice to follow a link. We also left the author notes section blank since it is not used by the Browser.

There is a one-to-one correspondence between links in an article and the file name lists. For example, if the second link in an article points to a file called CONTRNTS.FIL, then the second file name in the list of article files will be CONTENT. Picture links are implemented in the same way.

3.2. System Files

There are four system files, each with its own format:

TITLE.TIE: This file contains the title page for an encyclopedia along with the time and date of creation.

INDEX.TIE: The index file contain an alphabetized listing of article names, the corresponding file names, and all synonyms.

PICINDEX.TIE: The picture index file contains an alphabetized listing of picture file names and the corresponding synonyms.

REFER.TIE: The reference file was not implemented (or analyzed) because it is not used by the Browser. However, it s probably used by the citations portion of the Author.

3.3. File that were not Automatically Generated

The program that We developed generates all files used by the Browser except TITLE.TIE, MAINMENU.FLE, and HOWTOUSE.FLE. We chose to create these manually since they remained static regardless of how the manual was divided. We also did not create REFER.TIE since it is not used by the Browser.

4. Pass 2

The purpose of Pass 2 was to build a list of links for Pass 3. This pass was needed because Pass 3 would be inserting links to sections it had not reached, and Pass 3 therefore could not know the names of the sections. The following information was sent to the link file:

r	the reading number
n	the nthsection of the reading
x	the xthforced article in the section
P	page number(for manual double-checking)
*	division indicator
A	the article name

This file was then used by Pass 3 to create a table with the "previous", "review", and "next" links indicated.

5. Pass 3

Pass 3 created an encylopedia that is directly browsable by Hyperties and a list of figures to be be scanned by graphics scanner.

5.1. Table of Contents

Besides creating the reading articles, the converter also builds a table of contents and individual reading menus. The table of contents uses indentation and spacing to differentiate between chapter / section / subsection divisions and indicates the reading number for each division. Divisions that are article names(i.e. they are the first division in an article) are linked to the corresponding articles.

5.2. Reading Menus

Reading menus contain division names for a particular reading. Presenting reading menus with indentation levels (the way the table of contents was created) would be misleading in many cases. A subsection may not actually be "under" the section it appears to be since divisions from other readings are a not included. Instead, we used asterisks to indicate the division level. As in the table of contents, links are created from division headings that are article names.

5.3. Article Linking

All articles have links to the Main Menu, Contents, and the corresponding reading menu. This allows the user the browse the encyclopedia in several ways and to get an overview of the encyclopedia. Explicit links in the AIX manual were represented as previous, review and next links. The links were placed at the beginning of each article to allow the user to scan through articles. The links were also placed at the end of articles, so that the user could continue with another article without paging to the beginning of the article.

5.4. Algorithm

The following is an overview of the algorithm used in Pass 3.

Notes:

1. Flags shown are set when the line is read in and reset (acknowledged) when the condition is handled.
2. DividerName is the chapter / section / subsection heading.
3. The NewReading indicator is set when a reading ends, and superfluous text is skipped until a divider is reached.
4. Case is implemented as if ....then.....else.

```
{ * Overview of Pass 3 algorithm * }
Create the article index from the link file
Skip to the start of the first reading and get first non-blank line
while the index has not been reached
  do case
      case: NewReading and Divider
            Create a new article with the beginning links
            Send the DividerName to the article
            Send the DivderName to the Contents and
                   ReadingMenu as a link
      case: EndOfReading or Force
            Output the end links to the article and close
      case: Figure
            Output figure number and page number to figure list
            Output picture link to the article
      case: Divider
            Send DividerName to the article, Contents, and ReadingMenu
      case: not NewReading
            if line is "This chapter / section describes..."
                then Skip to next divider
                else Send the current paragraph to the article
    end case
    Get next non-blank line
end while
Create the picture index
End
```

Ⅳ.Conclusions

The main goal of this project was to test the feasibility of automatically converting a Linear Paper-Based Document into hypertext. Most of the problems associated with the conversion of the AIX manual were problems or limitations of the selected target hypertext system. Without the limitation on the number of articles, nodes could easily have been created from the table of contents. Also, given into-links, could have been forged linearly from the table of contents and non-linearly from the index and internal references (e.g. see topic on page number). However, this solution does not address two major problems: node size and non-explicit links.

Using the solution described above, nodes are created without consideration to the amount of information they contain. Nodes may be created which contain too much information. For example, a very large node could be subdivided so that it is a skeleton of the original node with links to the details or related information. On the other hand, if the table of contents is very detailed, the user may be overwhelmed by the task of constantly following links while staying oriented within the system.

Thus, node size is a problem in automatic hypertext construction. However, it is inherent to hypertext construction in general, not just automatic construction.

Also, Non-explicit links are those links which are not easily detectable without natural language understanding at a level that is currently unavailable on computers. The problem of extracting details from a node and creating several interconnected nodes is an example.

Some types of links will always require intelligent human linkers and may never be created automatically. This does not detract from automatic hypertext construction since automation, in any area, is restricted to tasks which do not require expertise to perform.

Using simple methods, such as described above, many classes of Linear Paper-Based Documents can be automatically converted. The resulting document may then be "fine-tuned" initially by a professional linker and then by user over the course of time.

With very large documents, such as manuals, there is a need to be able to convert the document directly into a form that is usable by a particular hypertext system. Manually inserting a large number of links which are regular in nature invites error and frustration.

It is unlikely that the software developer would be willing to provide a detailed description of the storage system used for several reasons. First, the software is probably a product that is being marketed. Revealing the storage system would allow competitors to create compatible products which bypass the supplier's product completely.

Also, many of the storage systems are quite complicated. Thus, it would be difficult for a user to write the necessary application software to convert their linear document into a hypertext document. Even if the user could write the software, it would probably be unwise to allow users to directly manipulate the storage system.

On the other hand, a supplier could not foresee all the possible input forms of a document or the "best" node/link layout. It would therefore be unreasonable to expect the supplier to write the software to convert any and all documents to a corresponding hypertext form.

Instead, a supplier might include a conversion program which would take in a standard (and easily describable) form of a document with the nodes and links somehow denoted and convert this intermadiate form directly to a hypertext document. Then the user would only have to write the software to convert a document to this simplified format.

V. Summary

As more documentation is put on-line, the demand for quick information-retrieval systems will increase. Hypertext is emerging as a storage and retrieval systerm that may satisfy this demand.

If hypertext proves to be a good model for this area, then a large body of information which already exists in paper form will need to be converted to hypertext. The AIX project showed that for an example manual on a selected system, this conversion is possible, and reasonable, using simple keyword matching algorithms.

The method used for this conversion does not address the issues of node size (a general hypertext problem) or non-explicit links (a natural language problem). For conversion to be practical on complicated system, suppliers will need to provide the user with intermediate conversion software.

Acknowlegement

This study was financially supported by a central research fund in 1995 from PaiChai university.

References

1. Bush, Vannevar, "As We May Think," Atlantic Monthly, No. 176, July 1945, pp.101-108.
2. Hyperties User's Manual, First Edition, Cognetics, Corporation, Princeton Junction, New Jersey, July 1987.
3. Conklin, Jeff, "Hypertext: An Introduction and Survey," IEEE Computer, Vol.20, No.9, pp.17-41.
4. Delisle, Norman, Schwartz, Mayer, "Neptune: a Hypertext System for CAD Applications," Proceedings of the ACM SIGMOD, International Conference on Management of Data, Washington D.C., 1986, pp.132-143.
5. Engelbart, Douglas, Augmenting the Human Intellect: A Conceptual Framework, Stanford Research Institute, Menlo Park, California, October, 1962.
6. Garrett, L. Nancy, Smith, Karen, Meyrowitz, Norman,"Intermedia: Issues, Strategies, and Tactics in the Design of a Hypertext Document System, "Proceedings of the Conferenc on Computer-Supported Coopera-tive Work, MCC Software Technology Program, Austin, Texas, 1986.
7. Halasz, Frank, Thomas P.Moran, Randall H. Trigg, "NoteCards in a Nutshell," Proceeding of the ACM Confernce on Human Factors in Computing Systems, Toronto, Canada, April, 1987.
8. A Guide to the Tagged Image File Format, Hewlett-Packard Company Greeley, Colorado, 1987.
9. System Product Interpreter User's Guide, Release 3, International Business Machines Corporation, Austin, Texas,1984.