PDF2XML: Converting PDF to XML

Yonggao Yang Kwang Paick Yanxiong Peng Yukong Zhang
Department of Computer Science
Prairie View A&M University
Prairie View, TX 77446

Abstract: *XML is a markup language for documents containing structured information. It is designed to make it easy to interchange structured documents over the Internet and further integrate them with management database system. PDF is a document format intended to electronically reproduce the look of a page. There is a huge demand of converting existing PDF documents into XML documents, so that they will be searchable and manageable. Since PDF is basically a page layout format and does not carry original document structure, converting PDF to XML remains a challenging task. This paper addresses the related technique problems and explores approaches. As part of the Data Conversion Project under development at the Data Conversion Center funded by DoD, we present a system, PDF2XML, designed to automatically perform the conversion with minimum human interaction.*

Keywords: Data Conversion, PDF, XML

1. Introduction

*P*ortable *D*ocument *F*ormat, PDF, is a widely used electronic document format intended to electronically reproduce the look of a page [1, 2]. It was designed to be a publishing format, which is often referred to as "Electronic Paper". However, a PDF document does not contain any structure information that its original electronic file possesses, such as paragraph information, table structures, and so forth. This allows a PDF document to be usually much smaller in size than its original electronic file in other formats, thus makes PDF documents convenient for transmitting on the Internet.

*EX*tensible *M*arkup *Language*, XML, is a markup language for documents containing structured information [3, 4, 5]. It allows that richly structured documents can represent database data as well as other kinds of structured data, and be used over the web in communication between business applications. XML provides a facility to define tags and the structural relationships between them.

Huge amount of documents are generated everyday and stored in PDF format. Furthermore, we have very large amount of legacy PDF documents without their original electronic files from which they were generated. Virk [8] lists the gains we can achieve by converting other documents into XML format. It might be easy to perform some conversions among various formats, such as XML to PDF and HTML to XML [6, 7].

Currently most of the conversion work is done manually, thus is time-consuming and error-prone. Few efforts are directed towards designing automatic conversion tools. Ouahid and Karmouch [6] address methods of converting web pages into XML documents. Youn and Ku [7] discuss issues of migrating data from one system to another, without mentioning PDF-to-XML. One commercial software worth of mention is Omnimark [9], which is designed to help convert documents in RTF format into XML format.

We are aware of that it is extremely difficult to develop a system to automatically conduct the 100% of PDF to XML conversion due to the versatility of PDF documents and XML DTD specification. We lunched a project to develop a computer system capable of performing the 60~70% of

the conversion task automatically, thus leaving about 30% of conversion task to be done manually. As the result of this project, a system, PDF2XML, was developed.

2. Convert PDF to XML

Our goal is to design and implement a system to help convert PDF document into XML documents with minimum human interaction. The structure of the system, PDF2XML, is depicted in Figure 1. The filled thick arrows show the flow of the data from the input of the PDF document to the output of the XML document. It goes through four phases: (1) PDF Extraction; (2) Structure Reconstruction; (3) Manual Modification; and (4) XML Wrapping.

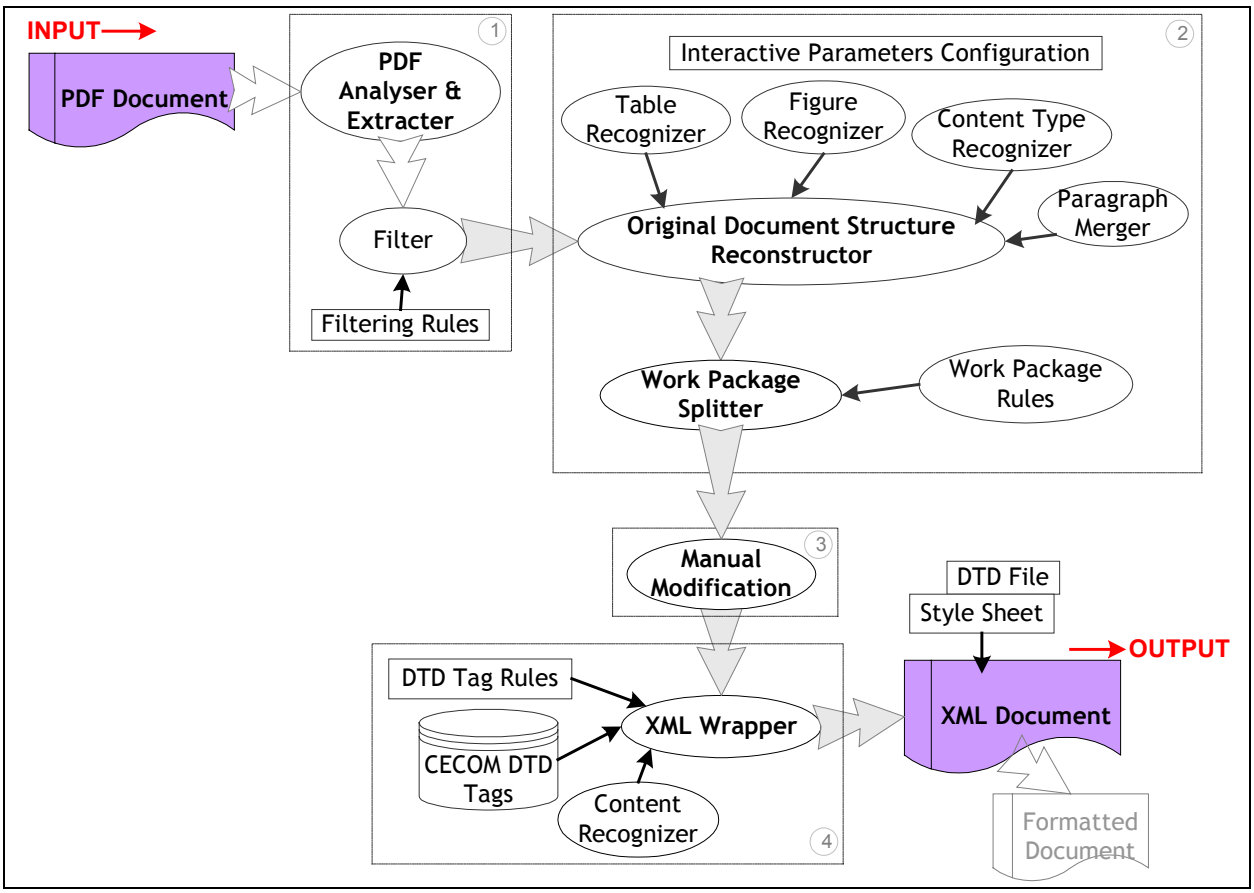
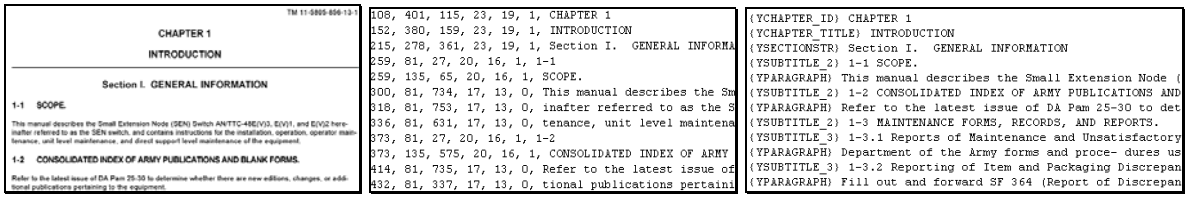


Figure 1: PDF2XML system structure

2.1 Phase-I: Information Extraction from PDF Document

The first phase is to extract as much information as possible from the PDF document, and filter out those unnecessary contents, and save it as a text file. A PDF Normal document does not contain original document structure. It is basically a page layout format, which shows where on the page a line should be placed. The information we can extract includes: (1) Document information: title, author, and date; (2) Page information: page number, page height and weight; (3) Each line string in the page: position information (top, left, width, and height), font information (size, type, style, and color), and others.

Figure 2 is an example that shows part of a sample PDF page and the extracted information. The numbers of each line in Figure 2(b) are Top, Left, Width, Height, Font Size, and Font Style (Other information is ignored because they are not required by the XML DTD system in this example).



(a) Part of a page in PDF (b) Extracted information in plain text (c) Output of Phase-II

Figure 2: Extract information from PDF

The filter and the filtering rules in this phase are used to throw away those non-required contents on PDF pages, such as the page numbers usually printed at the bottom of the pages, the document ID appearing on each page (e.g. the ID is at top-right corner in Figure 2(a)), and so forth.

2.2 Phase-II: Reconstruction of Original Document Structure

Reconstructing the original document structure from PDF file is the most challenging task of data conversion tools because PDF document does not carry any such information. The output of this phase is also a plain text file (Figure 2c).

Paragraph Merger: PDF document does not carry the paragraph information (Figure 2a). After the extraction, each line is an independent entity (Figure 2b). By using the line position, font, page number, and other parameters (such as space between regular lines and space between paragraphs, etc.), Paragraph Merger is able to tell the start and the end of a paragraph, and merge lines into paragraphs.

Figure Recognizer: XML treats each figure as an entity. Currently we manually snapshot each figure from PDF file and save it as a file, and simply use XML tags to wrap the figure's title to point to the file. However, during the automatic conversion, PDF2XML still needs to ignore the words belonging to the figure and extract only the figure title.

Table Recognizer and table structure reconstruction: Identifying tables and reconstructing their structures is the most difficult task for data conversion tools, particularly for those complex and irregular tables. Based on the position information and parameters provided by users, PDF2XML currently is capable of handling regular tables. However, for irregular tables, PDF2XML still relies on human interaction (Phase-III) to recover their structures.

Title Identifier: Identifying chapter titles, multi-level subtitles and footnotes is a critical task of reconstructing original document structure. Title Identifier of PDF2XML uses font size, font style, and line position, along with other parameters provided by the users to accomplish this task.

Figure 2c shows partial of the output from this phase. We use several predefined pseudo-tags (in curly braces at the beginning of each line) to mark the types of the contents. The major pseudo-tags are {CHAPTER_ID}, {CHAPTER_TITLE}, {SECTIONSTR}, {SUBTITLE_*x*}, {PARAGRAPH}, {FIGURE}, {TABLE}, and others. Here {SUBTITLE_*x*} means a level *x* subtitle.

2.3 Phase-III: Manual Modification

After Phase-II, we almost recover and reconstruct the original document structure from the PDF document, except those special cases that the previous two phases are not able to handle. This phase allows us to use any text-editing tool to modify the output of Phase II.

2.4 Phase-IV: XML Wrapper

After Phase-III, the remaining task is to use XML DTD tags to wrap the contents in the plain text file accordingly, and to generate the XML document. This task involves XML DTD tags and their syntax (rules), and original document structures stored in the plain text file from Phase 3. PDF2XML uses an Access database system to store all the DTD tags and the rules regarding the use of them. Users should modify this DTD database to meet their own XML DTD requirements. Sometimes, what DTD tags we should use depends on the content (we name it content-sensitive tags), "Content Recognizer" module is called to parse the content and make a decision on this. For some special cases, users might

want to identify this manually and insert specified control marks appropriately to the plain text file to facilitate XML Wrapper doing its job.

Figure 3 is a snapshot of PDF2XML interface. The top panel is used to provide system-use-help information, view the source PDF file, display and edit the multiple temporary files generated during the several conversion phases, and display the generated XML file with or without its style sheet. The bottom panel hosts all the control-buttons and various parameter-editing boxes. Through this panel, users interact with PDF2XML.

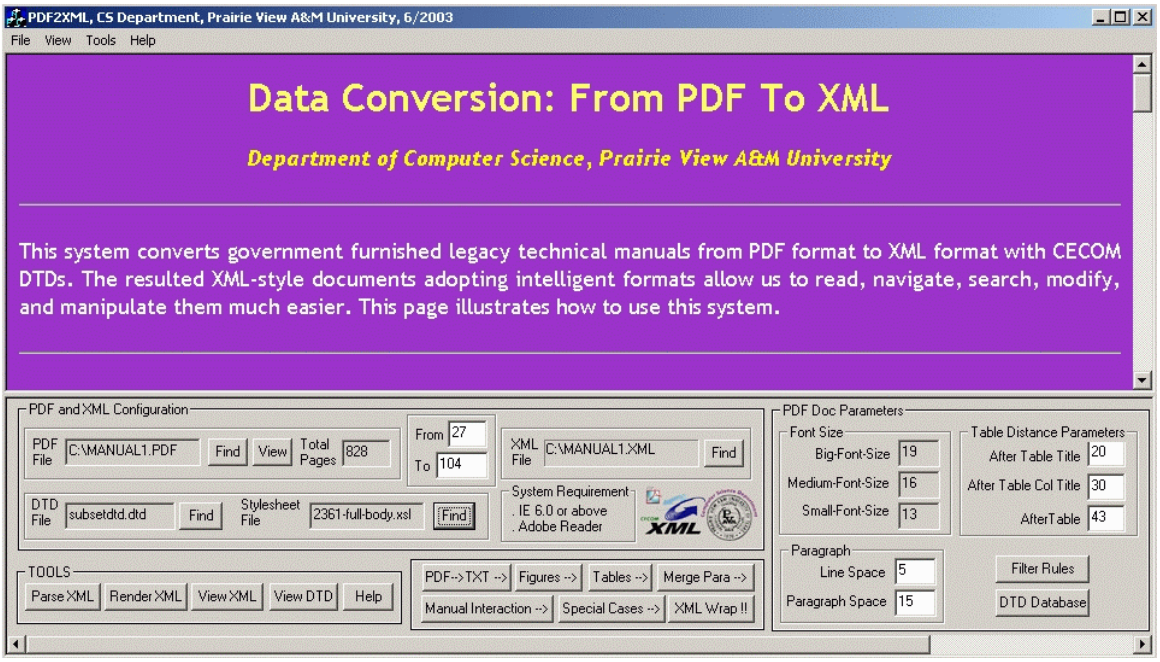


Figure 3: A snapshot of the PDF2XML interface

PDF2XML system was implemented with Visual C++ on Windows environments. It uses Access as the database system to store and locate CECOM DTD tags and wrapping rules.

3. Conclusion

Currently, by using PDF2XML, we are working with DoD to convert huge amount of weapon and equipment manuals in PDF format to XML documents. We plan to continue improving PDF2XML system and integrate other functions, including converting figures in PDF files to SVG Format that can be integrated into XML documents.

References

[1] Inc. Adobe Systems, "PDF Reference: Version 1.4," *Addison-Wesley Pub Co.*, 2001.

[2] Inc. Adobe Systems, "PDF Reader," http://www.adobe.com

[3] XML official website, http://www.xml.org

[4] Coyle, F. P., "XML, Web Services, and the Data Revolution," *Addison-Wesley Pub Co.*, 2002.

[5] Birbeck M., etc., "Professional XML," *Wrox Press Inc.*, 2000.

[6] Ouahid, H., and Karmouch, A., "Converting Web Pages into Well-formated XML Documents," *IEEE Proc. of International Conference on Communications*, 1999, pp. 676~680.

[7] Youn, C., and Ku, C. S., "Data Migration," *IEEE Proc. of the Fifth Distributed Memory Computing Conference*, 1992, pp. 1028~1037.

[8] Virk, R, "Why Use XML for Documents & Content?" http://www.datawarehouse.com/iknowledge/whitepapers/CID3443.pdf

[9] Barker, M., "Internet Programming with OmniMark," *Kluwer Academic Pub.*, 2000.