Converting PDF to HTML approach based on Text Detection

Deliang JIANG
College of Computer Science and Technology Zhejiang University
YuQuan Campus, HangZhou, China
+86 (0)571 87934822
jiangdl@zju.edu.cn

Xiaohu YANG
College of Computer Science and Technology, ZheJiang University
YuQuan Campus, HangZhou, China
+86 (0)571 87934822
yangxh@zju.edu.cn

ABSTRACT

Converting PDF document to HTML document with the same layout format is a very important and interesting research problem. After the conversion, it is easy for PDF document to be browsed online and information extracted. Based on the extraction result of the PDF document of the open source tool PDFBox, the paper described a method that can detect the layout information of the PDF document and convert the PDF document to HTML page effectively.

Keywords

PDF; Document conversion; Layout

1. INTRODUCTION

PDF----Portable Document Format is a very useful document format, it is independent of the operating system and can be viewed on any computer. It has played an important role in the information storage and transmission in many companies. PDF is based on a page description language [7]. Therefore it doesn't contain the information about the data structure of the file content, which restricts the information extraction and document conversion of the PDF file. Moreover, many documents are published online at present and the main online browsing format is the HTML format. Although the PDF files have the high compression feature, the HTML file is significantly smaller then the equivalent PDF content [2]. And in the traditional Content Management System, the management and publication of the content is always executed by different person. The author firstly finishes writing the content and save it as some file format such as the WORD or PDF, then uploads this file to the CMS system. After receiving the file, the administrator will extract the content in the file and publish it on the internet with the same layout. If the CMS system can convert the PDF file to be HTML file with the same layout automatically, it will alleviate the related workload and improve the efficiency.

The motivation of this work stems from document conversion on the Web. The method presented in this paper, which can be used to convert PDF into HTML with the same layout and font size effectively.

2. Related Work

The key of converting PDF to HTML is to recognize the various segments on the page firstly, and then extract the information from the segment. Finally, do the related conversion procedure. Many approaches have been put forward to extract the segment information in web pages [6], scanned images of pages [5] and monospaced terminal documents [4]. These methods can not be used in PDF file directly. And in [3], the author put forward a segmentation algorithm to extract segment information. But the approach is mainly worked on a binaries bitmap image, rather than a high level representation of the document such as the PDF file as input [1]. In [2], the author designed an algorithm to recognize the text segments of the PDF file; it is based on the vertical gap detection method. This method can be used in single structure PDF file effectively, but the impact on complex structure PDF file is not satisfactory. Based on the vertical gap detection method, the paper described an improved algorithm called as Text Segment Detection to recognize the text segments. It will detect the vertical and horizontal empty texts area firstly, and then combine these locations' information to recognize the text segments. Finally do the conversion procedure segment by segment.

3. Extracting Data from PDF

Because PDFBox Java library can parse the low-level data from the PDF file [8]. It encapsulates all the operations of accessing the PDF document page and parsing the instructions. We use PDFBox to generate text and graphic objects for each PDF instruction, each with its respective coordinates and attributes. We call these text and graphic objects as text fragments. Their information is stored in a structure named as *TextPosition*. Its fields are described as the following:

- **x:** The x coordinate of the text fragment.
- **y:** The y coordinate of the text fragment.
- **width:** The width of the text fragment.
- **height:** The height of the text fragment.
- **character:** The character to be displayed.
- **font:** The current font for this text fragment.

- **fontsize:** The font size of the text fragment.

Once we have got that information, we can do the conversion procedure by using the algorithm described in the following section. That information is important and will be fully utilized during all the stages, especially in the Text Segment Detection stage.

4. PDF to HTML Conversion

4.1 Design Target

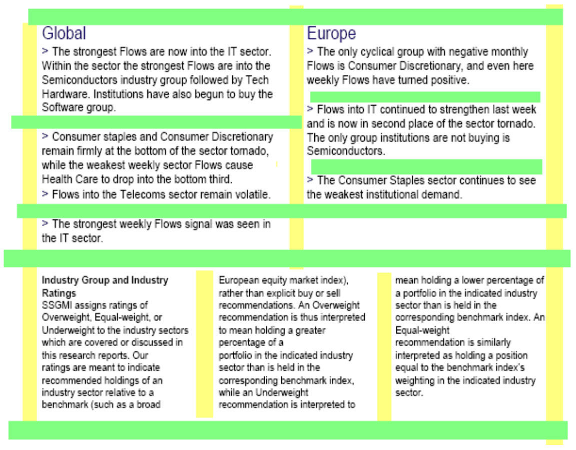
Firstly, we design to use PDFBox to extract all the text fragments and then sort them in some order (see section 4.2). Secondly, for single column PDF file, we consider all the text fragments belong to one text segment and do the text fragment merge process (see section 4.3) for this segment. As for multi column PDF file, because the reading order of the text fragments is not as same as the physical order, we design to find all the text segments by using the segment detection algorithm (see section 4.2) above all, and then do the text fragment merge process for each detected segment. Finally, the whole PDF file will be converted to be HTML file as the format described in figure1.

```
<HTML>
<HEAD>
………
</HEAD>
<BODY>
<STYLE type="text/css">
   <!--
               .ft1{font-size:11px;font-family:Helvetica; }
               ……
           .ftn{font-size:16px;font-family:Helvetica;}
           ……
   -->
</STYLE>
<DIV style="position:absolute;top:35;left:44">
       <DIV style="position:relative;top:10;left:20">
              <span class="ft1"> TEXT1</span>
       </DIV>
</DIV>
……
<DIV style="position:absolute;top:136;left:44">
       <DIV style="position:relative;top:55;left:78">
              <span class="ftn"> TEXTn</span>
       </DIV>
</DIV>
……
</BODY>
     </HTML>
```

Figure 1. Final HTML output format.

4.2 Text Segment Detection

For the single column article, we think that all the text fragments are belonged to one segment. As for the multi column article, first of all, we will find the horizontal and vertical empty rectangle text areas separately as shown in figure2. All the green shadow rectangle areas represent the horizontal empty rectangle text areas. All the yellow shadow rectangle areas represent the vertical empty rectangle text areas.


Global

> The strongest Flows are now into the IT sector. Within the sector the strongest Flows are into the Semiconductors industry group followed by Tech Hardware. Institutions have also begun to buy the Software group.

> Consumer staples and Consumer Discretionary remain firmly at the bottom of the sector tornado, while the weakest weekly sector Flows cause Health Care to drop into the bottom third.

> Flows into the Telecoms sector remain volatile.

> The strongest weekly Flows signal was seen in the IT sector.

Europe

> The only cyclical group with negative monthly Flows is Consumer Discretionary, and even here weekly Flows have turned positive.

> Flows into IT continued to strengthen last week and is now in second place of the sector tornado. The only group institutions are not buying is Semiconductors.

> The Consumer Staples sector continues to see the weakest institutional demand.

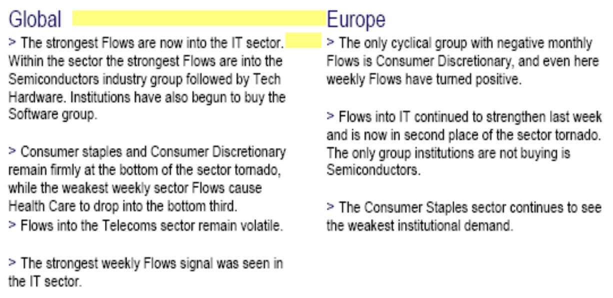
Industry Group and Industry Ratings
SSGMI assigns ratings of Overweight, Equal-weight, or Underweight to the industry sectors which are covered or discussed in this research reports. Our ratings are meant to indicate recommended holdings of an industry sector relative to a benchmark (such as a broad European equity market index), rather than explicit buy or sell recommendations. An Overweight recommendation is thus interpreted to mean holding a greater percentage of a portfolio in the indicated industry sector than is held in the corresponding benchmark index, while an Underweight recommendation is interpreted to mean holding a lower percentage of a portfolio in the indicated industry sector than is held in the corresponding benchmark index. An Equal-weight recommendation is similarly interpreted as holding a position equal to the benchmark index's weighting in the indicated industry sector.

Figure 2. The shadow areas represent the empty text area

Secondly, locate all the text segments through those empty text areas' coordinates. We create a class named as *Rectangle* to hold the coordinates of the bounding boxes of the empty text area. The *Rectangle* has four fields (*lowerLeftX, lowerLeftY, upperRightX, upperRightY*) to record the empty rectangle's diagonal coordinates. The segment detection algorithm is an improved algorithm based on the gap detection algorithm [2] put forwarded by *Tamir Hassan*. It is described as following:

Text fragments Sort. For vertical empty rectangle text area detection, we will sort the text fragments according to y-coordinates by ascending order. If the y-coordinates are equivalent, then we will sort the text fragments according to x-coordinates by ascending order. On the other hand, for horizontal empty rectangle text area detection, we will sort the text fragments according to x-coordinates by ascending order. If the x-coordinates are equivalent, then we will sort the text fragments according to y-coordinates by ascending order.

Vertical empty text rectangle area detection. Assume there exists two successive fragments *TextFragment1* and *TextFragment2* (*TextFragment1.X* < *TextFragment2.X*). And they are on the same line (*TextFragment1.Y* ≈*TextFragment2.Y*) in original article. If the distance between these two fragments is lager than the average distance among all the fragments of the same line, then we consider that there exists an empty text rectangle area between the two fragments (see figure3). We create an instance of the class *Rectangle* to store the empty rectangle area's coordinates.


Global

> The strongest Flows are now into the IT sector. Within the sector the strongest Flows are into the Semiconductors industry group followed by Tech Hardware. Institutions have also begun to buy the Software group.

> Consumer staples and Consumer Discretionary remain firmly at the bottom of the sector tornado, while the weakest weekly sector Flows cause Health Care to drop into the bottom third.

> Flows into the Telecoms sector remain volatile.

> The strongest weekly Flows signal was seen in the IT sector.

Europe

> The only cyclical group with negative monthly Flows is Consumer Discretionary, and even here weekly Flows have turned positive.

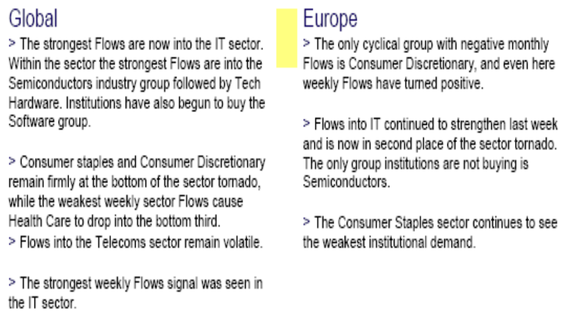
> Flows into IT continued to strengthen last week and is now in second place of the sector tornado. The only group institutions are not buying is Semiconductors.

> The Consumer Staples sector continues to see the weakest institutional demand.

Figure 3. The shadow areas represent the empty text rectangle area of two successive fragments

Whenever a new empty area is found, it should be compared to those of the rectangles that have been created already in order to decide whether we should create a new rectangle or not. If the new empty area intersects with the original empty area, there is no need to create a new rectangle instance to represent the new empty text area. What we should do is just to stretch the original rectangle vertically or reduce the original rectangle horizontally according to the comparison rules (see figure4).The rules are described as following:

- If the horizontal coordinates were intersected each other or identical between the new empty text area and the original empty text area, the original one should stretch vertically.
- If the horizontal coordinate were partly intersected those created rectangles, the original empty text rectangle area should be reduced horizontally
- When the empty text rectangle area was totally intersected by a text block, it should be marked as "closed" and should not be compared with the following new founded empty text area(see figure5).



Global

> The strongest Flows are now into the IT sector. Within the sector the strongest Flows are into the Semiconductors industry group followed by Tech Hardware. Institutions have also begun to buy the Software group.

> Consumer staples and Consumer Discretionary remain firmly at the bottom of the sector tornado, while the weakest weekly sector Flows cause Health Care to drop into the bottom third.
> Flows into the Telecoms sector remain volatile.

> The strongest weekly Flows signal was seen in the IT sector.

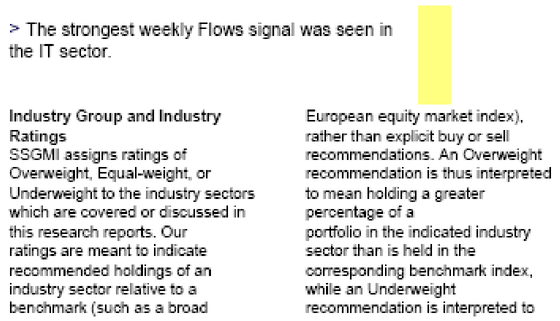
Europe

> The only cyclical group with negative monthly Flows is Consumer Discretionary, and even here weekly Flows have turned positive.

> Flows into IT continued to strengthen last week and is now in second place of the sector tornado. The only group institutions are not buying is Semiconductors.

> The Consumer Staples sector continues to see the weakest institutional demand.

Figure4. The modified empty text rectangle area after being stretched vertically and reduced horizontally



> The strongest weekly Flows signal was seen in the IT sector.

Industry Group and Industry Ratings
SSGMI assigns ratings of Overweight, Equal-weight, or Underweight to the industry sectors which are covered or discussed in this research reports. Our ratings are meant to indicate recommended holdings of an industry sector relative to a benchmark (such as a broad

European equity market index), rather than explicit buy or sell recommendations. An Overweight recommendation is thus interpreted to mean holding a greater percentage of a portfolio in the indicated industry sector than is held in the corresponding benchmark index, while an Underweight recommendation is interpreted to

Figure5. the "closed" empty text rectangle area

Horizontal empty text rectangle area detection. This algorithm is similar with the detection method of vertical empty text rectangle area. Assume there exist two successive fragments *TextFragment*1 and *TextFragment2* (*TextFragment*1.Y < *TextFragment2.Y*) and they belong to the same column (*TextFragment1.X* ≈*TextFragment2.X*) in original article. If the distance between these two fragments is lager than the average distance among all the fragments of the same column, then we consider that there exists an empty text rectangle area between the two fragments. After finding the new empty text rectangle area, the empty text rectangle area should be compared to those existing rectangles to judge whether we should create a new rectangle or not. The comparison rules is similar with the vertical empty area detection rules except that we should compare the vertical coordinate instead of the horizontal coordinate.

Text segments detection. After we have found all the vertical and horizontal empty text rectangle areas, we can detect the location of the text segment by computing the coordinate of the empty rectangles. This procedure can be divided into the following four steps:

a) Iterate the vertical empty text rectangle. For each vertical rectangle----*currentVerticalRectanglle*, compare its coordinate with the horizontal empty text rectangle----*currentHorizontalRectangle* one by one.

b) If *currentVerticalRectangle* intersect with the *currentHorizontalRectangle*, then we should find the next vertical empty text rectangle which intersect with *currentHorizontalRectangle*, too. If found, we named the rectangle as *nextVerticalRectangle*.

c) If the *nextVerticalRectangle* exists, then we should iterate the horizontal empty text rectangle to find the rectangle that *intersect* with both *nextVertiaclRectangle* and *currentVerticalRectangle*. If found, we named the rectangle as *nextHorizontalRectangle*.

d) When the *currentVerticalRectanglle*, *currentHorizontalRecttangle*, *nextVerticalRectangle* and *nextHorizontalRectangle* have been found, we consider the area they surround is the location of the text segment. We create a instance to store the coordinate information of the text segment:

- lowerLeftX=currentVerticalRectangle.upperRightX
- lowerLeftY=currentHorizontalRectangle.upperRightY
- upperRightX=nextVerticalRectangle.lowerLeftX
- upperRightY= nextHorizontalRectangle.lowerLeftY

4.3 Text merge algorithm

After having detected all the text segments, in the first instance, we should find out all the text fragments belonging to each text segment. In the next place, for each segment, do the text merge process in order to convert the whole texts to be the HTML format described in figure1. The process of text merge algorithm is described as the following:

Assume there exists two successive text fragments $TextFragment_{i-1}$ and $TextFragment_i$; the segment's location information is stored in $DIVRectangle_j$; the merge result is stored in an instance of string called as *output*. First of all, we should judge that whether $TextFragment_{i-1}$ and $TextFragment_i$ are in the same line or not. If they are in the same line, we should use rule **a)** to merge this two text fragments, otherwise, we should use rule **b)** to merge them.

a) If the font information of $TextFragment_{i-1}$ and $TextFragment_i$ is equivalent. Then we just connect these two fragments' texts:

$$output = output + TextFragment_i.Text$$

If the font information of this two fragments is not equivaient, then it means that the two fragments can not use he same style tag, we should add a new style node st_i to the CSS style sheet and add a closed tag </span> to the *TextFragment*$_{i-1}$:

.st$_i${*font-size: TextFragment*$_i$*. fontsize;font-family: TextFragment*$_i$ *.font*}

output = output +</span> + <span class="st$_i$*">+ TextFragment*$_i$*.Text*

b) Because this rule is used in the situation that *TextFragment*$_{i-1}$ and *TextFragment*$_i$ are not in the same line, we should add the closed tag </span></DIV> to the *TextFragment*$_{i-1}$ to show that current line has been processed completely. As for the *TextFragment*$_i$, firstly, we should compute its relative coordinate (*relativeX*, *relativeY*) accroding to the *DIVRectangle*$_j$. Secondly, create the tag *<DIV style = “position:relative top:relativeY; left:relativeX”>* to express that a new line process has started. Finally, add a new style node *st*$_i$ to the CSS style sheet to represent the font information of the *TextFragment*$_i$:

.st$_i$ {*font-size:TextFragment*$_i$*.fontsize;font-family: Text-Fragment*$_i$ *.font*}

output = output+</span></DIV> + <DIV style = “positi-on:relative;top: relativeY;left:relativeX"> + <span class="st$_i$*">+ TextFragment*$_i$*.Text*

After all the above procedure, all the texts’ mergence and conversion has been finished. The final step of the algorithm is to put the processed texts into *DIVRectangle*$_j$:

output = <DIV style="position:absolute; top: DIVRectangle$_j$*.lowerLeftY; left: DIVRectangle.lowerLeftX; width: upperRightX-lowerLeftX; height:upperRightY-lowerLeftY; "> + output + </DIV>*

5. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a novel method that can convert the PDF file to be the HTML file with the same layout and font information. We put forward an algorithm that can recognize the text segments in PDF file. We presented the steps of merging text fragments. We utilize the coordinate information obtained from the open source tool PDFBox to do the conversion. Experiments showed that the method is effective. In our next stage work, we will mainly focus on the image conversion from PDF file to HTML file. In additional, we try to convert more font information such as font color, font type, etc.

6. REFERENCES

[1] Tamir Hassan and Pobert Baumgartner, “Table Recognition and Understanding from PDF Files,” IEEE Computer Society, ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02 , Vol02, Sep. 2007

[2] Tamir Hassan, PDF to HTML Conversion. DOI = http://www.dbai.tuwien.ac.at/staff/hassan/pdf2html/final.pdf.

[3] Aiello M, Monz C, Todoran L and Worring M, Document Understanding for a Broad Class of Documents. In Intl. J. of Doc. Anal. and Recog. (2002) 5(1) 1–16

[4] Kieninger T, Table Structure Recognition Based on Robust Block Segmentation. In Proc. of Document Recognition V. (1998) 22–32

[5] Ishtani Y, Fume K and Sumita K, Table Structure Analysis Based on Cell Classification and Cell Modification for XML Document Transformation. In Proc. of 8th Intl. Conf. on Doc. Anal. and Recog. (2005) (2) 1247–1252

[6] Gatterbauer W and Bohunsky P, Table Extraction Using Spatial Reasoning on the CSS2 Visual Box Model. Proc. of 21st Nat. Conf. on Artif. Intel. (2006)

[7] PDF Reference1.7., DOI = http://www.adobe.com/devnet/acrobat/pdf-

[8] Ben Litchfield, Making PDFs Portable:Integrating and Java Technology, DOI = http://pdfhome.hope.com.cn/Article.aspx?CID=bf51a5b6-78a5-4fa3-9310-16e04aee8c78&AID=168c3168-989a-4f1a-a801-76a8cfb64b59