**A REPORT**

**ON**

**Web Based Portal to Extract Text Data from Documents**

**By**

ID No.

2014A7PS026G

Nidhi Kadkol

**AT**

**Indira Gandhi Centre For Atomic Research**

**A Practice School I Station of**

**BIRLA INSTITUE OF TECHNOLOGY AND SCIENCE, PILANI**

**(June,2016)**

**Abstract**

This project is mainly concerned with extraction of useful data from documents.

Data is available in documents like word, PDF, web pages etc. Extraction of such data is essential to prepare keyword, metadata and data models for information retrieval.

The project is development of web portal to extract data from PDF files using algorithms. PDF files use compression and coding while it is prepared. The reverse has to be done. Also routines to convert all documents to PDF, so that with this portal extraction from all documents become possible. The extracted data has to be made into a useful data model.

This Project is halfway through completion, and at this point extraction of data from PDF(Portable Document Format) files, RTF(Rich Text Format) files, and extraction of metadata from images has been successful.

The language used for all these conversions is PHP.

This Project Report aims to help the reader understand the different aspects of PDF to text conversion, RTF to text conversion, and the implementation.

**Contents**

**Topic**

1. Introduction
2. Rich Text Format (RTF)
3. Portable Document Format (PDF)
4. Steps Followed
5. Implementation
6. Further Work
7. References

**Introduction**

Text is the main form of communicating knowledge. We use the term *document* to denote a single unit of information, typically text in a digital form, but it can also include other media. In practice, a document is loosely defined.

The syntax of a document can express structure, presentation style, semantics, or even external actions. The style of a document defines how the document is visualized in a computer window or a printed page.

Metadata: Most documents and text collections have associated with them what is known as metadata. Metadata is information on the organization of data, the various data domians, and the relationship between them. In short, metadata is ‘data about the data’.

Descriptive Metadata is metadata that is external to the meaning of the document, and pertains more to how it was created. Another type of metadata characterizes the subject matter that can be found within the document’s contents. This is known as Semantic Metadata.

With the advent of the computer, it was necessary to code text in binary digits. The first coding schemes were EBCDIC and ASCII, which used seven bits to code each possible symbol. Later, ASCII was standardized to eight bits (ISO-Latin), to accommodate several languages, including accents and other diacritical marks. Nevertheless, ASCII is not suitable for oriental languages such as Chinese or Japanese Kanji, where each symbol might represent a concept and therefore thousands of them exist. For this case, a 16-bit code exists called Unicode.

There is no single format for a text document, and an IR system should be able to retrieve information from many of them, In the past, IR systems would convert a document to an internal format. However, that had many disadvantages, because the original application related to the document is not useful any more. On top of that, we cannot change the contents of a document. Other text formats were developed for document interchange. The Rich Text Format (RTF) is used by word processors and has ASCII syntax. Other important formats were developed for displaying or printing documents. The most popular ones are the Portable Document Format (PDF) and Postscript (which is a powerful programming language for drawing).

**Rich Text Format (RTF)**

An RTF file consists of unformatted text, control words and groups.

A control word is a specially formatted command that RTF uses to mark printer control codes and information that applications use to manage documents. A control word is made up of lowercase alphabetic characters between "a" and "z". Each control word begins with a backslash (\) and ends with one of the following:

* A space. In this case, the space is part of the control word.
* A numeric parameter that can be a positive or a negative number.
* Any character other than a letter or a digit.

Therefore, the character string \rtf1\ansi\ansicpg1251 easily can be divided into three control words: rtf with parameter 1 (the major format version), ansi (the current encoding) and ansicpg with parameter 1251 (the current code page number 1251).

A group consists of text and control words enclosed in braces ({}). Control words defined within a group affect only the text inside this group and all nested subgroups. In order to know which control words are active now we will use the control words stack. When reading the opening brace ({) we will add new array stack element and write the data from previous stack element to it. When reading the closing brace (}) – remove top stack element.

Some control words may be turned off not by closing the group but adding parameter 0 to the control word. For example, strings This is {\b bold} text and This is \b bold \b0 text give us the same result: This is bold text.

Now we can come to the conclusion that all characters in an RTF file that are not control words or braces are plain text.

Keeping this in mind, we write the code to extract clean text from RTF files.

**Scope for Improvement:**

* We can add the additional checks for non-textual data. Given code will cut off only the fonts, colors, theme design, binary data and everything that is marked as "do not read me if you cannot" (text marked with \ \*).
* We may add the further encoding and code page parsing to reflect the \ 'hh keywords more accurately.

**Portable Document Format (PDF)**

PDF implements documents as a hierarchy of tagged objects organized into trees and/or linked lists. The objects can encapsulate various types of content, or attributes, or pointers to external resources.

There are eight basic kinds of objects in PDF: Booleans, numbers, names, strings, arrays, dictionaries, streams and the null object.

Strings

In PDF a string consists of a series of 8-bit bytes surrounded by parentheses. A string can be divided into several lines by using the backslash (\) at the end of the line. The backslash itself is not considered as part of the string. For example:

( This is a string. )  
( This is a longer \  
string. )

Any 8-bit value can be represented either by its octal equivalent (in the form \ddd, where ddd is the octal number), or by its two-digit hex equivalent, surrounded by angle brackets. Later we will search for the text data in the strings.

Arrays

An array is a sequence of PDF objects, enclosed in square brackets. For example:

[(Hello,)10(world!)]

Dictionaries

A dictionary is the key/value pairs, enclosed in two left angle brackets (<<) in the beginning and two right angle brackets (>>) at the end:

<< /Length 4 0 R  
   /Filter /FlateDecode  
>>

A dictionary is used to assign some properties to an object. We will use these data to determine how to decrypt the stream, find its length, or, for example, omit the current object (if it is an image).

Streams

A stream is a sequence of 8-bit bytes between the keywords stream and endstream. Any type of content made up of raw binary data is represented by a stream.

Streams are represented as objects (see below), which also means the stream will be bracketed by obj and endobj keywords. Before the stream keyword there must be a stream attribute dictionary, giving information about stream length (/Length key) and, often, the kind of compression employed (/Filter key).

As an example, a small text stream might look like:

2 0 obj  
<<  
/Length 39  
>>  
stream  
BT  
/F1 12 Tf  
72 712 Td (A short text stream.) Tj  
ET  
endstream  
endobj

In this example, the text itself is given as a string followed by the display text operator Tj.

Objects

An object can enclose the content of any PDF data types (Boolean, number, name, string, etc.), bracketed between obj and endobj keywords. We are primarily interested in objects with the streams inside.

**To get clean text:**

To look for text objects in a PDF-document, we look for objects that contain streams.

The text in a stream is enclosed between BT (beginning of text) and ET (end of text) keywords.

PDF displays a text if there is Tj (display text) or TJ (display text considering the individual character positioning) keyword after a text string or an array of strings.

* PDF supports the individual character positioning. This means that we can set arbitrary and individual size of the distance between each pair of characters.
* PDF supports composite fonts where a single character is encoded by one or more bytes of the string. In this case the code lengths and the mappings from codes to glyphs are defined in a data structure called a CMap. PDF also uses a special ToUnicode CMaps to map character codes to Unicode values.

**Steps Followed**

* Read the data from pdf file
* Get all text data.
* Get the list of all objects.
* Select objects with streams.
* Check if an object includes data stream.
* Check object parameters and look for text data.
* Decode text data

**Implementation**

To implement the extraction, the code was written in PHP language.

PHP stands for Hypertext Preprocessor and is a server-side language. When a visitor opens the page, the server processes the PHP commands and then sends the results to the visitor's browser. A typical PHP file will cause commands to be executed in the server in addition to the usual mixture of text and HTML (Hypertext Markup Language) tags. PHP is Open Source and cross-platform. PHP runs on Windows NT and many Unix versions, and it can be built as an Apache module and as a binary that can run as a CGI.

PHP's language syntax is similar to C's and Perl's. Variables don't have to be declared before usage, and it's easy to create arrays and hashes (associative arrays). PHP even has some rudimentary object-oriented features.

**Function calls for converting PDF to Text**

function pdf2text($filename) {   
  
    // Read the data from pdf file  
    $infile = @file\_get\_contents($filename, FILE\_BINARY);   
    if (empty($infile))   
        return "";   
  
    // Get all text data.  
    $transformations = array();   
    $texts = array();   
  
    // Get the list of all objects.  
    preg\_match\_all("#obj(.\*)endobj#ismU", $infile, $objects);   
    $objects = @$objects[1];   
  
    // Select objects with streams.  
    for ($i = 0; $i < count($objects); $i++) {   
        $currentObject = $objects[$i];   
  
        // Check if an object includes data stream.  
        if (preg\_match("#stream(.\*)endstream#ismU", $currentObject, $stream)) {   
            $stream = ltrim($stream[1]);   
  
            // Check object parameters and look for text data.   
            $options = getObjectOptions($currentObject);   
            if (!(empty($options["Length1"]) && empty($options["Type"]) && empty($options["Subtype"])))   
                continue;   
  
            // So, we have text data. Decode it.  
            $data = getDecodedStream($stream, $options);    
            if (strlen($data)) {   
                if (preg\_match\_all("#BT(.\*)ET#ismU", $data, $textContainers)) {   
                    $textContainers = @$textContainers[1];   
                    getDirtyTexts($texts, $textContainers);   
                } else   
                    getCharTransformations($transformations, $data);   
            }   
        }   
  
    }   
    // Analyze text blocks taking into account character transformations and return results.   
    return getTextUsingTransformations($texts, $transformations);

This code will parse correctly the simple PDF files. It can be used as a basis and improved it according to specific needs of an organization.

**Further Work**

Extraction of data from Word Files, as well as web pages. Word Files have the *.docx* format, and Web Pages are HTML. Converting Web Pages to PDF can also be used and an intermediate step to extract clean text from Web Pages.

Furthermore, extraction of structured text from PDFs and web pages will be useful in searching for keywords in documents.

**References**

* webcheatsheet.com
* Ricardo Baeza-Yates and Berthier Ribiero-Neto, *Modern Information Retrieval*
* latestcode.net