Generating Summary Documents for a Variable-Quality PDF Document Collection

Jacob Hughes
School of Computer Science
University of Nottingham
NOTTINGHAM NG8 1BB, UK
jxh00u@cs.nott.ac.uk

David F. Brailsford
School of Computer Science
University of Nottingham
NOTTINGHAM NG8 1BB, UK
dfb@cs.nott.ac.uk

Steven R. Bagley
School of Computer Science
University of Nottingham
NOTTINGHAM NG8 1BB, UK
srb@cs.nott.ac.uk

Clive E. Adams
Institute of Mental Health
University of Nottingham
NOTTINGHAM NG7 2TU, UK
clive.adams@nottingham.ac.uk

ABSTRACT
The Cochrane Schizophrenia Group's Register of studies details all aspects of the effects of treating people with schizophrenia. It has been gathered over the last 20 years and consists of around 20,000 documents, overwhelmingly in PDF. Document collections of this sort – on a given theme but gathered from a wide range of sources – will generally have huge variability in the quality of the PDF, particularly with respect to the key property of text searchability.

Summarising the results from the best of these papers, to allow evidence-based health care decision making, has so far been done by manually creating a summary document, starting from a visual inspection of the relevant PDF file. This labour-intensive process has resulted, to date, in only 4,000 of the papers being summarised – with enormous duplication of effort and with many issues around the validity and reliability of the data extraction.

This paper describes a pilot project to provide a computer-assisted framework in which any of the PDF documents could be searched for the occurrence of some 8,000 keywords and key phrases. Once keyword tagging has been completed the framework assists in the generation of a standard summary document, thereby greatly speeding up the production of these summaries. Early examples of the framework are described and its capabilities illustrated.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Search Process, Selection Process; I.7.5 [**Document Capture**]: Optical Character Recognition (OCR)

General Terms
Algorithms, Documentation, Languages

Keywords
Schizophrenia; PDF; OCR; document collections

FINAL DRAFT of Short Paper accepted for
DocEng'14, September 16–19, 2014, Fort Collins, CO, USA.
Copyright is held by the owner/author(s). Published version is at
http://dx.doi.org/10.1145/2644866.2644892

1. INTRODUCTION
In the field of medicine it is rarely the case that a single set of trials can give a definitive answer about the efficacy of the treatment, or treatments, under evaluation. A UK epidemiologist called Archie Cochrane was one of the first to call for collecting all controlled trials of health care and for combining results from similar, suitably rigorous, trials, In this way one can deliver the most accurate estimate of the effects of care.

Cochrane's influence led to the Cochrane Database of Systematic Reviews and, in the case of schizophrenia, a repository collated over 20 years, comprising six decades of published evidence – now held at the Institute of Mental Health at the University of Nottingham [1]. A problem with combining several studies, however well conducted they may have been, is that there is little chance that results are recorded in a standard way, let alone any possibility of access to a project's internal documentation to acquire results directly. The 20,000 documents of the Cochrane Schizophrenia Group's Register of Trials is overwhelmingly in the form of published papers archived in Adobe's PDF. This format is far from ideal in terms of easy data extraction but, being ubiquitous, it is in reality 'the only show in town'.

To make full use of these PDF files it is necessary to manually extract all quantitative and qualitative data on methods, participants, interventions and outcomes into a standardized format. Until now this has required the effort of experienced researchers, often with a suitable medical background. Only 20% of the total number of trials have been data extracted – most data remain unused and potentially useful evidence of the effects of care are not fully utilized. Furthermore those data that have been extracted are often impossible to verify, since their exact origin within the original document is not transparent.

This paper describes a pilot project, representing the first stages of a concerted effort to provide a computer-assisted framework for tagging key phrases within this variable-quality PDF document collection, followed by automated extraction and assemblage of the tagged phrases into standardised summary documents.

2. OVERVIEW OF PDF
In October 1993 Adobe Systems Inc. introduced the Portable Document Format (PDF) and released viewer software for that format called Acrobat, initially for Macintosh and MS-Windows systems. PDF is an optimized development, (with document portability in mind) of the earlier PostScript format that had revolutionized typesetting in the 1980s.

The acceptance of PDF for providing distributable 'electronic page masters' was very rapid in the technical publication field.

Publishers soon became used to turning their PostScript files (used for creating the hard-copy form of their journals) into the equivalent PDF files, using Adobe's own Distiller conversion program. Acceptance of PDF took rather longer to happen in fields such as law, business and medicine. Here the problems included the legal acceptability of PDF master files (as opposed to traditional 'hard copy') coupled with waiting patiently for software such as word processors and spreadsheets to be able to export output to PDF swiftly and efficiently. However, by 2005, the status of PDF as an archival medium was becoming sufficiently clear for ISO to begin work on making PDF be an archival standard (PDF/A).

2.1 Varieties of PDF

PDF offers a convenient way of making high-quality documents be readily exchangeable. Ideally all the running text will be formatted in the chosen body-text typefaces; lettering within diagrams may be set in some different face. Line diagrams will be drawn using the correct line-drawing primitives, while bitmapped material such as photographs (either lossily or losslessly compressed) is catered for by the PostScript/PDF **image** operator. PDF files of this quality are referred to as PDF-FTG (i.e. 'PDF, Formatted Text and Graphics')[1]. The key advantage of this format is that text strings can usually be located within the PDF of the body text.

It would be comforting to hope that all collected corpora of PDF documents would be of PDF-FTG quality. More often than not this fails to be achieved because the 'umbrella' nature of PDF allows scanned-page documentation to be stored in PDF **image** (PDF-I) format which can replicate, inside a PDF wrapper, widely accepted image formats such as JPG or TIFF. However, this PDF-I format, just like JPG and TIFF, is not text-searchable.

In 1994 Adobe introduced an OCR-based product called Acrobat Capture. When applied to a PDF-I file it went beyond mere OCR by creating an invisible, searchable, text overlay, using what was technically called 'Text Mode 3' (see [2] page 306). By suitably adjusting the point sizes and the inter-word spacing in this hidden textual layer it was possible to make it be in exact registration with the perceived words in the page image. Adobe named this new hybrid format 'PDF Image plus Hidden Text' (PDF-IT). It has the great virtue that searched-for words are highlighted via the correct bounding box in the textual layer but they show up as highlights in the exactly superposed image layer, thereby creating the illusion of a textually searchable image.

For more than ten years the technology from Acrobat Capture has been deployed in the full releases of Acrobat itself to satisfy a demand for making potentially huge document-bases of scanned material (e.g. in legal work) be text searchable.

3. FIRST STEPS

The long-term aim of this work is to fully realize the potential of the schizophrenia register of trials in terms of extracting 'best practice' procedures from sets of separate, but related, trials. In moving from a fully manual scheme to a computer assisted one it was immediately clear that it was vital to assess the relative proportions of PDF-FTG, PDF-I and PDF-IT files within the schizophrenia document-base

[1] In earlier documentation Adobe referred to PDF files of this sort as 'PDF Normal'

3.1 Analyzing the documents

An analysis took place in early December 2013 of 17,990 documents available to us from the document set. The analysis program was written in Objective C and used the Quartzcore Framework to provide PDF analysis functionality. The analysis algorithm relies on a function called `getPageType` which tests to see if there is a whole-page image layer on the current page but with no associated textual content stream; if so then the page is Image Only. The second per-page test is to see if the word count for invisibly-rendered text exceeds that for visibly rendered. If the majority of the text is invisible, in most of the pages, then the document is PDF-IT. Otherwise it is a PDF-FTG file.

Having applied the above algorithm to the full set of 17,990 PDF files the following breakdown of the document set was obtained (actual numbers of files are in parentheses)

• PDF-FTG	62.8%	(11,296)
• PDF-I	32.2%	(5,796)
• PDF-IT	5.0%	(898)

3.2 Converting PDF-I to PDF-IT

By applying the OCR functionality of the Acrobat Capture plugin, with the batch processing capability contained in the full Acrobat product, from Acrobat 7.0 onwards, it was possible to automate the conversion of the 5796 PDF-I files into the PDF-IT format. Details of the required procedure are given on an Adobe Web site [3]. Preliminary tests showed that, on average, each of these file conversions took just over 2 minutes using Acrobat X Pro running on a 2.4 GHz Pentium system. This led to an estimate of 8 days to complete the conversion task — an estimate which turned out to be remarkably accurate.

Given that almost a third of the document collection was PDF-I, and hence not text-searchable, it was a vital first step, in contemplating automated keyword recognition for the whole document collection, to establish that this retrofitting of searchability was indeed feasible.

4. SYSTEMATIC REVIEWS

The Cochrane Group currently organises a large number of medical professionals to undertake systematic reviews of health care topics based on reports of trials. A systematic review attempts to identify, appraise and incorporate all relevant evidence that meets pre-specified criteria, in order to answer a given research question. Data harvested from the trials, as recorded in the PDF, are manually entered into a separate system called RevMan [4]. This software – a writing and analysis tool – helps prepare data for meta-analysis and publication.

The current process of data input for systematic reviewing is as follows:

1. Read through the PDF for the chosen clinical trial.
2. Identify a piece of data (quantitative or qualitative) within the PDF, that is relevant for extraction.
3. Manually input this data into RevMan

An important requirement here is for total accuracy in identifying and transferring information in steps 2 and 3. Furthermore, for data-checking at a later stage, the page number and location of all data items should be logged as part of stage 3. It is precisely these two activities that could benefit most from an automated system.

Note also that the conversion of PDF-I files to PDF-IT, described in the previous section, is yet another big bonus for transferring data into RevMan, because cut-and-paste of text from a PDF file is not possible for PDF-I – in this case the reviewer would have to manually type all of the relevant data into RevMan.

5. HIGHLIGHTS AND LINKS IN PDF FILES

PDF itself defines various ways in which selected objects such as images, text, section headings and so on can be highlighted or made the target of hyperlinks either internally (e.g. from a table of contents) or externally from an incoming link. This latter often originates in HTML but can finish on a fixed destination page within the PDF file using the so-called Named Destinations facility.

These features are challenging indeed to add, programmatically, into arbitrary PDF material of variable quality, as encountered in the current studies. Nevertheless Adobe's (very costly) Acrobat SDK will enable a JavaScript programmer to add exactly these features, thereby enhancing the original PDF. We shall call this a *direct* editing system because it alters the original PDF file.

For all the above reasons of technical difficulty, third-party PDF analysis and augmentation systems commonly display page images of the PDF and create the illusion of genuine PDF highlights by holding the PDF's text, in HTML, alongside the page image. Any requested annotations or highlighting are superposed on the page image using a variety of ingenious methods, often known generically as *standoff markup* [5], for cross linking the HTML to the image. This is an example of an *indirect* editing system, where the PDF file itself is not altered.

In wanting to create an environment for automating the highlighting of key phrases, and for extracting data from our schizophrenia documents, we looked at a few indirect systems already available, such as Document Cloud and ExaCt. Both of these provided some aspects of the facilities we were seeking but neither of them provided all we wanted. For this reason we chose to implement our own system.

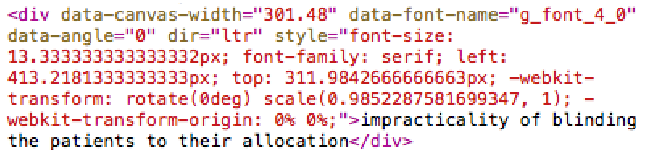
6. A PROTOTYPE SYSTEM IN PDF.js

Part of the core functionality required for our text highlighting and extraction system (PET) was the ability to interact with a PDF document;. An Open Source library called *PDF.js* [6] has been found that provides this functionality by creating a platform on which an interaction API can be developed.

Our prototype PDF extraction tool (PET) was implemented within the PDF.js library, initially created by Mozilla (and which is still in the early stages of development). PDF.js renders PDF files to a native HTML format using Javascript.

A large proportion of this HTML version of the PDF is constructed of HTML `DIV` tags, which are used to denote a division or section within an HTML file. Each `DIV` contains a text area that has been transformed to align with the original PDF's text. In this sense it resembles the approach taken by the hidden text layer within a PDF-IT file.

Below is an example of what a small segment of this HTML code looks like when displaying a few lines of text in a PDF file, using the PDF.js library.



```
<div data-canvas-width="301.48" data-font-name="g_font_4_0"
data-angle="0" dir="ltr" style="font-size:
13.333333333333332px; font-family: serif; left:
413.2181333333333px; top: 311.9842666666663px; -webkit-
transform: rotate(0deg) scale(0.9852287581699347, 1); -
webkit-transform-origin: 0% 0%;">impracticality of blinding
the patients to their allocation</div>
```

Figure 1. PDF text as an HTML <DIV> within PDF.js

One of the problems posed by PDF.js is that it does not support the display of some extra objects, optionally present in a PDF file, such as Named Destinations, Article Threads and highlighted text. It also does not provide the ability to 'write back' data into the PDF source file, to enhance the richness of the original document.

Despite the fact that PET had to be prototyped as an indirect editing system we were fully aware that any such system has the huge drawback that any enhancements to the PDF file are visible only within the indirect framework (DocumentCloud, ExaCT, PET etc.). Ultimately this means that anyone wanting to see the PDF enhancements has to install a copy of the indirect system.

The first requirement of the PET functionality was to identify, and link back to, the page on which a particular keyword occurred. It was already known that this could be achieved in PDF itself by a facility known as Named Destinations, held within the root node of the PDF tree. Equally, PDF itself implements its own highlighting facility (located on the actual PDF pages) which can easily reproduce any indirect highlights added within PET.

If we could find a way to make PET be a more direct system and to overwrite the source PDF file with Named Destinations and highlights then we would achieve the goal of making the summary XML document, and its corresponding enhanced PDF, be a free-standing pair of documents – not at all dependent on the PET framework for visual display, nor for acting as an independent resource for further research.

6.1 Text highlighting in PET

So far we have described how an experienced researcher might transfer tagged phrases from a PDF file into RevMan either by cut-and-paste or via direct text input. The PET environment provides automated assistance by allowing a researcher to highlight key phrases in the PDF image, by simply dragging the mouse over them. Further assistance is at hand in the shape of a dataset of 8,000 keywords and key phrases (painstakingly built up over 20 years). This keyword dataset takes the form of row entries in a spreadsheet file that can readily be converted into database entries in some suitable format.

The architecture of PET is based around communicating client and server processes. The front-end client allows for the display of highlighted tagged phrases either manually or automatically generated and for later editing, or even complete removal, of the highlighted phrase. The server process handles the generation of a suitable subset of keywords from the keyword database and it also copes with writing out the final XML summary document, together with a revised version of the input PDF. This revised version of the PDF uses Named Destinations to implement a bounding box and a page number for each highlighted keyword.

6.2 Auto-generation of highlighted tags

The main driver for generating highlighted tags for keywords was the dataset of keywords and key phrases. To speed up the tagging process this dataset was used, first of all, to search the PDF file under analysis, very quickly, for whether any of these words or phrases actually occurred. At this stage no attempt was made to find the page number, or page position, of each occurrence. Once the 'used subset' of the 8,000 keywords had been found, the second pass of the tagging process could start to create tags that include page position and page number.

We should note, at this stage, that problems sometimes occurred with the PDF file's textual content, which is contained within HTML `DIV` markers inside the PET/PDF.js environment. The text blocks on a PDF page are not always rendered to the screen in what the user might think of as "correct reading order". For this reason, when PDF.js converts the text streams of the PDF into HTML DIVs, the reading order may not look correct and this can be particularly problematical if a key phrase straddles more than one DIV. The problem is neatly solved by being prepared to shuffle the DIVs around into a correct reading order, as determined by their textual placement co-ordinates.

A sample of highlighted text, as it appears in PET, and corresponding to the coding in Figure 1, is shown below

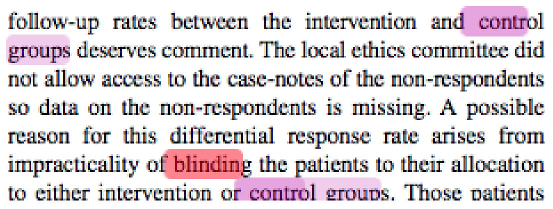

follow-up rates between the intervention and control groups deserves comment. The local ethics committee did not allow access to the case-notes of the non-respondents so data on the non-respondents is missing. A possible reason for this differential response rate arises from impracticality of blinding the patients to their allocation to either intervention or control groups. Those patients

Figure 2. Highlighting and tagging of keywords

At the end of the tagging process (which may involve a mixture of auto-generated and manually added tags) an XML-structured file of these tags is handed back from client to server as shown in the sample below.

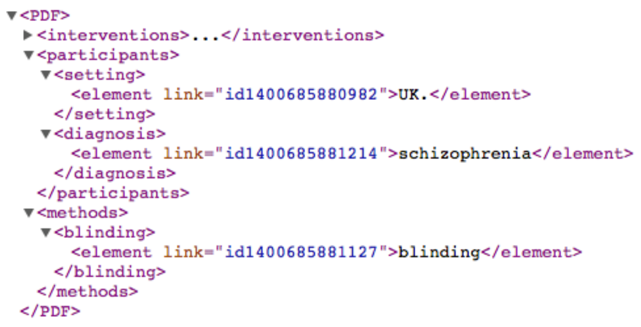

```
▼<PDF>
 ▶<interventions>...</interventions>
 ▼<participants>
  ▼<setting>
     <element link="id1400685880982">UK.</element>
   </setting>
  ▼<diagnosis>
     <element link="id1400685881214">schizophrenia</element>
   </diagnosis>
  </participants>
 ▼<methods>
  ▼<blinding>
     <element link="id1400685881127">blinding</element>
   </blinding>
  </methods>
 </PDF>
```

Figure 3. A portion of the XML summary document

Each individual data tag was assigned its own unique ID, generated using the exact date and time of its creation,. Using this ID it is possible to edit and remove tags either individually or in groups. The unique ID was also ideally suited for the generation of a unique name when each of the tags was converted into a Named Destination.

6.3 Implementation of Named Destinations

In addition to making the XML summary file of Figure 3 available as one of the standard outputs from PET there was also a need to write back an enhanced PDF file, ideally with a Named Destination for each tag and with PDF highlighting of those same tags. Given that we have used PDF.js and Python hosted on the cloud compute platform provided by Google, specifically GAE (the Google App Engine), to create an 'indirect' system, it comes as no surprise that it lacks the ability to alter the input PDF file by writing back an enhanced version of it. For this reason the server side of the PET system calls on a separate module, written in C, to perform this task. It takes as its input a comma-separated variable (CSV) list denoting each of the desired Named Destinations This module was written by one of us (SRB) using technology from the COGs [7] project.

A glance at page 82 of the PDF reference manual [2] shows that Named Destinations are located at the root of the PDF tree and are therefore relatively easy to insert. However, the related desire to make all these places be PDF highlights, to mirror those highlights we created in the PET environment (see Figure 2), is a much harder task. Highlights in PDF are located on the actual PDF pages. Some very careful calculations have to be performed to convert the bounding boxes of the PET highlights into PDF format, followed by equally careful tree traversal, inside the PDF, to plant the highlights in the correct format on the correct page.

7. DISCUSSION AND CONCLUSIONS

The work described here was a pilot project to try out an editing and creation environment (PET) for use by researchers when tagging and highlighting key words and phrases in a database of randomized trials. An important issue was whether, when tagging was complete, other medical sub-specialties could get benefit from this markup in the form of an enhanced PDF file and an XML summary document i.e. without having to install the entire PET environment.

The results we obtained are very encouraging. Much work remains to be done in reducing the number of 'hits' that result when the dataset of keywords is processed against the content of the target paper. A greater degree of context sensitivity is needed possibly coupled with a learning mechanism connected to gathering data about which of the auto-inserted key phrases the researcher chooses to accept or discard. As noted in the previous section there is also a large amount of work to be done in writing back the corrected highlights into the PDF file.

Encouraged by what has been achieved already a consortium of enthusiastic medical-informatics professionals has now submitted a major bid for European funding so that the full potential of the schizophrenia register of trials can be exploited.

8. REFERENCES

[1] C. E. Adams, The Cochrane Schizophrenia Group's Specialised Register of trials, 2014. http://szg.cochrane.org/cszg-specialised-register

[2] Adobe Systems Inc, *PDF Reference (Third Edition; PDF 1.4),* Addison Wesley, 2002.

[3] Rick Borstein, *Batch Processing using Acrobat Professional,* 2005. http://blogs.adobe.com/acrolaw/2005/10

[4] Cochrane Informatics and Knowledge Management Dept., *RevMan*,. http://tech.cochrane.org/Revman

[5] Peter L. Thomas and David F. Brailsford, "Enhancing Digital Documents using XML-based Standoff Markup", *Proceedings ACM Document Engineering Symposium (DocEng05),* pp. 177–186, Nov. 2005.

[6] Mozilla (Open Source), *PDF.js*. https://wiki.mozilla.org/PDF.js

[7] Steven Bagley, David Brailsford, and Matthew Hardy, "Creating well-structured PDF as a sequence of Component Object Graphic (COG) elements", *Proceedings ACM Document Engineering Symposium (DocEng03),* pp. 58–67, Nov. 2003.