

## Lab 6 (2 hours): Advanced Document Types

*This lab consists of a series of independent examples that deal with different document types. They don't need to be completed in order. There will not be time to complete all the exercises in this lab session. Please start with Parts I and II, then choose any other exercises that are relevant to the documents that you will be working with back home.*

*The labs are:*

*Part I – Scanned Images and OCR text*

*Part II – Adding scanned documents to your custom collection*

*Part III – Web sites and web pages*

*Part IV – Word documents*

*Part V – PDF and PowerPoint documents*

*Part VI – CDS/ISIS databases*

*Part VII – Your custom collection*

### Part I – Scanned Images and OCR text

---

*Here we build a small replica of Niupepa, the Maori Newspaper collection, using six newspapers taken from three newspaper series. It allows full text searching and browsing by title and date. When a newspaper is viewed, a preview image and its corresponding plain text are presented side by side, with a goto page navigation feature at the top of the page. The collection involves a mixture of plug-ins, classifiers, and format statements. The bulk of the work is done by PagedImgPlug, a plug-in designed precisely for the kind of data we have in this example. For each document, an “item” file is prepared that specifies a list of image files that constitute the document, tagged with their page number and (optionally) accompanied by a text file containing the machine-readable version of the image, which is used for full text searching. Four newspapers in our collection (three from the series Te Whetu o Te Tau, one from Te Haeata) have text representations, and two (from Te Waka o Te Iwi) have images only. Item files can also specify metadata. In our example the newspaper series is recorded as ex.Title and its date of publication as ex.Date. This metadata is extracted as part of the building process.*

1. Start a new collection called **Paged Images** and fill out the fields with appropriate information: it is a collection sourced from an excerpt of Niupepa documents; the only metadata used is document title and date, and these are extracted from the “item” files included in the source documents so no metadata set need be stipulated.
2. In the **Gather** panel, open the *niupepa\sample\_items* folder in *workshop\_files* and drag the two subfolders into your collection on the right-hand side. A popup window asks whether you want to add **PagedImgPlug** to the collection to process this file. Click **<Add Plugin>**, because this plugin will be needed to process the item files.
3. Some of the files you have just dragged in are the newspaper images, others are text files that contain the text extracted from these images. We want these to be processed

by **PagedImgPlug**, not **ImagePlug** or **TEXTPlug**. Switch to the **Design** panel and delete **ImagePlug** and **TEXTPlug**. While you are at it, you could tidy things up by deleting all plugins from **HTMLPlug** to **NULPlug** as well, since they will not be used.

4. Open up the configuration window for **PagedImgPlug** by double-clicking on the plugin. Switch on its **screenview** configuration option by checking the box. The source images we use were scanned at high resolution and are large files for a browser to download. The *screenview* option generates smaller screen-resolution images of each page when the collection is built.
5. Now go to the **Create** panel, **build** the collection and **preview** the result. Search for *waka* and view one of the titles listed (all three appear as *Te Whetu o Te Tau*). Browse by *titles a–z* and view one of the *Te Waka o Te Iwi* titles.

This collection was built with Greenstone's default settings. You can locate items of interest, but the information is less clearly and attractively presented than in the full Niupepa collection.

#### *Grouping documents by series title and displaying dates within each group*

6. Under *titles a–z* documents from the same series are repeated without any distinguishing features such as date. It would be better to group them by series title and display dates within each group. This can be accomplished using an **AZCompactList** classifier rather than **AZList**, and tuning the **VList** format statement.
7. In the **Design** panel, under the **Browsing Classifiers** section, delete the **AZList** classifiers for *ex.Source* and *ex.Title*.
8. Now add **AZCompactList** for *ex.Title* and **DateList** for *ex.Date*.
9. **Modify** the format statement for **VList** (under **Format Features**). Find the part of the default statement that says

```
{If}{[ex.Source],<br><i>([ex.Source])</i>}
```

and change it to

```
{If}{[ex.Date],: [ex.Date]}
```

Click **<Replace Format>**. This has the effect of displaying the extracted date information, if present.

10. **Build** and **preview** the collection.

### *Searching at page level*

11. The newspaper documents are split into sections, one per page. For large documents, it is useful to be able to search on sections rather than documents. This allows users to more easily locate the relevant information in the document.
12. Go to the **Search Indexes** section of the **Design** panel. Remove the **ex.Source** index. Select the **text** index in the **Assigned Indexes** box, and change the **Index Name** to “whole newspapers”. Click **<Replace Index>**. Create a new index: set the **Index Name** to “newspaper pages”, keep **text** selected in **Build index on**, and change **At the level** to **section**. Click **<Add Index>**. Click **<Set Default Index>** on the right hand side to make the ‘pages’ index the default.
13. **Build** and **preview** the collection. Compare searching in the “whole newspapers” index compared to the “newspaper pages” index. A useful search term for this collection is “aroha”.
14. You will notice that when searching for individual pages, the newspaper image is displayed in the search results. As these images are very large, this is not very useful. To remove this, edit the format statement for **VList** (under **Format Features**), and remove the second line:

```
<td valign=top>[ex.srclink]{Or}{[ex.thumbicon],  
[ex.srcicon]}[ex./srclink]</td>
```

**Preview** the collection—the search results should be back to normal.

### *Displaying scanned images*

15. When you reach a newspaper, only its associated text is displayed. When either of the *Te Waka o Te Iwi* newspapers is accessed, the document view presents the message *This document has no text*. No scanned image information (screen-view resolution or otherwise) is shown, even though it has been computed and stored with the document. This can be fixed by a format statement that modifies the default behaviour for **DocumentText**.
16. In the **Format Features** section of the **Design** panel, select the **DocumentText** format statement. The default HTML format string displays the document’s plain text. For documents which had no text, **PagedImgPlug** set their text to “This document has no text”.

Insert the following text after the `<tr>` and before the `<td>[Text]</td>`:

```
<td valign=top>[srclink] [screenicon] [/srclink]</td>
```

and click **<Replace Format>**. The resulting format statement will look like:

```
<center><table width=_pagewidth_><tr>
<td valign=top>[srclink] [screenicon] [/srclink] </td>
<td>[Text] </td></tr></table></center>
```

(This format statement can be copied and pasted from the file *workshop\_files\niupepa\doc\_tweak.txt*).

Including [screenicon] has the effect of embedding the screen-sized image generated by switching the **screenview** option on in **PagedImgPlug**. It is hyperlinked to the original image by the construct [srclink]...[/srclink].

17. Switch to the **Create** panel and **preview** the revised collection.

In the collection you have just built, newspapers are grouped by series title, and dates are supplied alongside each one to distinguish it from others in the same series. Users can browse chronologically by date, and when a newspaper page is viewed a preview image is shown on the left that displays the original high-resolution version when clicked, accompanied on the right by the plain-text version of that newspaper (if available).

*Adding another newspaper to the collection*

18. Another newspaper has been scanned and OCR'd, but has no item file. We will add this newspaper into the collection, and create an item file for it.

19. Go back to the **Gather** panel. Add the folder *workshop\_files\niupepa\new\_paper\12* to your collection.

A series of popups ask you about adding plugins to the collection to process the text and image files. Remember that ImagePlug and TextPlug were removed from the collection as we wanted these files to be processed by PagedImgPlug. Click **<Don't Add Plugin>** for each popup.

You may notice that for text files, GLI suggests **ProCitePlug** as the plugin to add. If you open up the **Select plugin to add** drop down list, you can see that **TEXTPlug** is also suggested. Both these plugins process files with extension .txt.

20. Inside the **12** folder you can see that there are 4 images and 4 text files.

21. Create an item file for the collection. Have a look at an existing item file to see the format. Start up **NotePad** (Start→Programs→Accessories→NotePad) to open a new document. Add some metadata. The **Title** for this newspaper is "Te Haeata 1859-1862". The **Date** is "18610902". (Greenstone's date format is yyyyymmdd.) Metadata must be added in the form:

```
<Metadata name>Metadata value
```

22. For each page, add a line in the file in the following format:

```
pagenum:imagefile:textfile::
```

For example, the first page entry would look like

```
1:images/12_3_6_1.gif:text/12_3_6_1.txt::
```

Note that if there is no text file, you can leave that space blank.

23. Save the file in **My Documents**, using **Filename** *12\_3\_6.item*, and **Save as type** *All files*. (Don't save as type .txt as this will save the file as 12\_3\_6.item.txt.) Back in the **Gather** panel of GLI, locate the new file in the **Workspace** tree (**Home Folder**→**My Documents**), and drag it into the collection, adding it to the **12** folder.

24. **Build** the collection and **preview**. Check that your new document has been added.

#### *Advanced customization (Optional)*

1. Make each bookshelf node in the **Title** classifier show how many entries it contains. In the **Format Features** section of the **Design** panel, select the **Title** classifier from the **Choose Feature** drop down list (*CL1 AZCompactList –metadata ex.Title*), and **VList** from the **Affected Component** list. Append the following:

```
{If}{[numleafdocs],<td><i>([numleafdocs])</i></td>}
```

Click **<Add Format>**, switch to the **Create** panel, and click **<Preview>** (no need to rebuild).

This revised format statement has the effect of specifying in brackets how many items are contained within a bookshelf. It works by exploiting the fact that only bookshelf icons define `[numleafdocs]` metadata.

By modifying **CL1VList** instead of **VList**, the change will only apply to the first classifier (Titles).

2. Search results, at the page level, only show the Title of the page (the page number), not the Title of the paper. Modify the format statement to show the paper title as well as the page number. In the **Format Features** section of the **Design** panel, select **Search** in **Choose Feature**, and **VList** in **Affected Component**.

The extracted Title for the current section is specified as `[ex.Title]` while the Title for the parent section is `[parent:ex.Title]`. Since the same **SearchVList** format statement is used when searching both whole newspapers and newspaper pages, we need to make sure it works in both cases.

Set the format statement to the following:

```
<td valign=top>[link] [icon] [/link]</td>
<td valign=top>
{If}{ [parent:ex.Title], [parent:ex.Title]: } [ex.Title]
<br><i>({Or}{ [parent:ex.Date], [ex.Date] })</i></td>
```

(The format statement can be copied and pasted from the file *workshop\_files\niupepa\search\_tweak.txt*).

The first line links to the document. The third line displays the parent Title if there is one, then the Title of the current page or document. The fourth line displays either the parent Date (in the case of pages) or the Date (in the case of documents), in italics (<i>..*</i>*</i>).

3. Change the document display to show a link to hide/show the text. Modify the DocumentText format statement to look like the following:

```
<center><table width=_pagewidth_><tr>
<td valign=top>[srclink] [screenicon] [/srclink]</td>
<td valign=top>
{If}{_cgiargshowtext_,
<a href='_gwcgi_?e=_cgiarge_&a=d&d=_cgiargd_'>Hide
text</a><br/>[Text],
<a href='_gwcgi_?e=_cgiarge_&a=d&d=_cgiargd_&showtext=1'>Show
text</a>}
</td></tr></table></center>
```

(The format statement can be copied and pasted from the file *workshop\_files\niupepa\doc\_switch.txt*).

This displays the screen icon linked to the full size version as before, with a “show text” link, which when clicked will display the Text of the page.

## **Part II – Your custom collection**

---

Scan and add your print documents to your own collection. You will need to create an item file for each document and use PagedImgPlug to process the item files. There is no OCR software installed in the lab. You can either add the documents with images only, or create some text files manually that contain a small part of the contents of the documents.

### Part III – Web sites and web pages

---

This exercise uses the **Download** panel in the Librarian Interface to download files and websites from the web. Then we modify the collection slightly to make Greenstone point back to these documents instead of displaying the local copies. Finally, we look at how to divide HTML documents into sections.

1. Start a new collection called **webtudor**, and base it on **New collection**.
2. In a web browser, visit <http://englishhistory.net>, follow the link to *Tudor England*, and click **<enter>**. You should be at the URL <http://englishhistory.net/tudor/contents.html>

This is where we started the downloading process to obtain the files you have been using for the **tudor** collection. You could do the same thing by copying this URL from the web browser, pasting it into the **Download** panel, and clicking the **<Download>** button. However, several megabytes will be downloaded, which might strain your network resources—or your patience! For a faster exercise we focus on a smaller section of the site.

3. In the **Download** panel, enter this URL <http://englishhistory.net/tudor/monarchs/edward6.html> into the **Source URL** box. There are several options that govern how the download process proceeds. To copy the *monarchs* section of the website, select **Only mirror files below this URL**. If you don't do this, the downloading process will follow links to other areas of the *englishhistory.net* website and grab those as well. Set **Download depth** to **Unlimited**—we want to follow as many links as necessary to download all the pages.
4. If your computer is behind a firewall or proxy server, you will need to edit the proxy settings in the Librarian Interface. Open the **Connection** tab in **File→Preferences...** and switch on the **Use proxy connection?** checkbox. Enter the proxy server address and port number in the **Proxy Host** and **Proxy Port** boxes. Click **<OK>**.
5. Now click **<Download>** on the **Download** panel. If you have set proxy information in Preferences, a popup will ask for your user name and password. Once the download has started, a progress bar appears in the lower half of the panel that reports on how the downloading process is doing.

More detailed information can be obtained by clicking **<View Log>**. The process can be paused and restarted as needed, or stopped altogether by clicking **<Close>**. Downloading can be a lengthy process involving multiple sites, and so Greenstone allows additional downloads to be queued up. When new URLs are pasted into the Source URL box and **<Download>** clicked, a new progress bar is appended to those already present in the lower half of the panel. When the currently active download item completes, the next is started automatically.

- Downloaded files are stored in a top-level folder called **Downloaded Files** that appears on the left-hand side of the **Gather** panel. You may not need all the downloaded files, and you choose which you want by dragging selected files from this folder over into the collection area on the right-hand side, just like we have done before when selecting data from the *workshop\_files* folder. In this example we will include everything that has been downloaded.

Select the **englishhistory.net** folder within **Downloaded Files** and drag it across into the collection area.

- Switch to the **Create** panel to **build** and **preview** the collection. It is smaller than the previous collection because we included only the *monarchs* files. However, these now represent the latest versions of the documents.

#### *Pointing to documents on the web*

- The files you have just downloaded were saved in a way that preserved the structure of the original site. This allows any page's original URL to be reconstructed from the folder hierarchy.
- In the **Design** panel, configure **HTMLPlug**. Switch on the **file\_is\_url** option. While you are there, switch off the **smart\_block** option so that the stray images are not processed.

Setting the **file\_is\_url** option to HTMLPlug means that Greenstone sets an additional piece of metadata for each document called URL, which gives its original URL.

It is important that the files gathered in the collection start with the web domain name (*englishhistory.net* in this case). The conversion process will not work if you dragged over a subfolder.

- To make use of the new URL metadata, the icon link must be changed to serve up the original URL rather than the copy stored in the digital library. In the **Design** panel, select the **Format Features** section and edit the **VList** format statement by replacing

```
[link][icon][link]
```

with

```
[weblink][webicon][weblink]
```

Click **<Replace Format>** to commit the change.

- Switch to the **Create** panel and **build** and **preview** the collection. Note that the document icons have changed. The collection behaves exactly as before, except that when you click a document icon your web browser retrieves the original document from the web. If you are working offline you will be unable to retrieve the document.



### *Extracting metadata*

12. Many HTML documents contain metadata in `<meta>` tags in the `<head>` of the page. Using WordPad, look at the file  
C:\Program Files\Greenstone\collect\webtudor\import\englishhistory.net\tudor\monarchs\boleyn.html.  
This page has *page\_topic*, *content* and *author* metadata.
13. Configure **HTMLPlug** to look for these metadata items. Switch on the **metadata\_fields** option, and set it to “Title,Author,Page\_topic,Content”.
14. **Build** the collection, then in the **Enrich** panel, look at the extracted for some of the HTML files in *englishhistory.net/tudor/monarchs*. The new metadata can now be used in classifiers or search indexes.

### *Section tagging for HTML documents*

1. In your digital library, take a look at the **Greenstone demo** collection. Browse to one of the documents. This collection is based on HTML files, but they appear structured in the collection. This is because these HTML files were tagged by hand into sections.
2. Using WordPad, open up one of the HTML files from the demo collection:  
C:\Program Files\Greenstone\collect\demo\import\fb33fe\fb33fe.htm. You will see some HTML comments which contain Section information for Greenstone. They look like:

```
<!--
<Section>
  <Description>
    <Metadata name="Title">Farming snails 1: Learning about snails;
Building a pen; Food and shelter plants</Metadata>
  </Description>
-->

<!--
</Section>
<Section>
  <Description>
    <Metadata name="Title">Dew and rain</Metadata>
  </Description>
-->
```

Where Greenstone encounters a `<Section>` tag in one of these comments, it will start a new subsection of the document. This will be closed when a `</Section>` tag is encountered. Metadata can also be added for each section—in this case, **Title** metadata has been added for each section. In the demo collection in the digital library, find the *Farming snails 1* document (through the *titles a-z* browser). Look at its table of contents and compare it to the Section tags in the HTML document.

3. Add a new Section into this document. For example, add a new subsection into the *Introduction* chapter. In **WordPad**, edit the html file, and add something like the following just after the Section tag for the Introduction section:

```
<!--  
<Section>  
  <Description>  
    <Metadata name="Title">Snails are good to eat. </Metadata>  
</Description>  
-->
```

Then just before the next section tag (*What do you need to start?*), add the following:

```
<!--  
</Section>  
-->
```

The effect of these changes is to make a new subsection inside the *Introduction* chapter.

4. Open the **Greenstone demo** collection in the Librarian Interface. In the **Document Plugins** section of the **Design** panel, note that **HTMLPlug** has the **description\_tags** option set. This option is needed when Section tags are used in the source documents.

The **metadata\_fields** option is not valid when **description\_tags** is set—all metadata is expected to be in the Section tags if they are being used.

5. **Build** and **preview** the collection. Look at the *Farming snails I* document again and check that your new section has been added.

## Part IV – Word documents

---

The standard way Greenstone processes Word documents is to convert them to HTML format using a third-party program, wwWare. This sometimes doesn't do a very good job of conversion. If you are using Windows, you can take advantage of Windows native scripting to do a better job of conversion. If the original document was hierarchically structured using Word styles, these can be used to structure the resulting HTML. Word document properties can also be extracted as metadata.

1. In your digital library, preview the **reports** collection. Look at the Word documents and notice how they have no structure—they have been converted to flat documents.

### *Using Windows native scripting*

2. In the Librarian Interface, open up the **reports** collection. Switch to the **Design** panel and select the **Document Plugins** section on the left-hand side. Double click the **WordPlug** plugin and switch on the **windows\_scripting** option.

3. **Build** and **preview** the collection. Have a look at word03.doc and word06.doc. These now appear with hierarchical structure. But these two are the only ones.

The default behaviour for WordPlug with windows\_scripting is to section the document based on “Heading 1”, “Heading 2”, “Heading 3” styles. If you open up the word03.doc or word06.doc documents in Word, you will see that the sections use these Heading styles.

Note, to view style information in Word, you can select **Format→Styles and Formatting** from the menu, and a side bar will appear on the right hand side. Click on a section heading and the formatting information will be displayed in this side bar.

4. Some of the documents do not use styles (e.g. word01.doc) and no structure can be extracted from them. Some documents use user-defined styles. WordPlug can be configured to use these styles instead of Heading 1, Heading 2 etc. Next we will configure WordPlug to use the styles found in word05.doc.

### *Defining styles*

5. Change the mode in the Librarian Interface to **Library Systems Specialist (File→Preferences→Mode)**. This is because you will need to use regular expressions to set up the style options.
6. In the **Document Plugins** section of the **Design** panel, select **WordPlug** and click **<Configure Plugin>**. Four types of header can be set which are:
  - title\_header (titleHeader1|titleHeader2|...)
  - level1\_header (level1Header1|level1Header2|...)
  - level2\_header (level2Header1|level2Header2|...)
  - level3\_header (level3Header1|level3Header2|...)

These header options define which styles should be considered as title, level 1, level 2 and level 3 styles. Open up the word05.doc in Word (by double-clicking on it in the Gather pane), and examine the title and section heading styles. You will see that various user-defined header styles are set such as:

- *PaperTitle*: Title of the paper
- *SammaryHeader* (probably mistyped): Summary section
- *ChapterTitle*: Level 1 section heading
- *SectionHeading*: Level 2 section heading
- *ReferenceHeading*: Reference section

Set the options in WordPlug as follows:

```
title_header: PaperTitle
level1_header: (SummaryHeader|ChapterTitle|ReferenceHeading|Reference_heading)
level2_header: SectionHeading
```

7. **Build** the collection and **preview** it. Look in particular at word05. You will see that this document is now also hierarchically structured.

#### *Removing pre-defined table of contents*

8. If you look at word06.doc you will see that it now has two tables of contents. One is generated by Greenstone based on the document's styles, the other was already defined in the Word document. WordPlug can be configured to remove predefined tables of contents and tables of figures. The tables must be defined with Word styles in order for this to work.
9. To remove the tables of contents and figures from word06.doc, switch on the **delete\_toc** option in WordPlug. Set the header styles as follows:

```
toc_header: (MsoToc1|MsoToc2|MsoToc3)
tof_header: MsoTof
```

Once these are set, click **&ltOK>**.

10. **Build** and **preview** the collection. word06.doc should now only have one table of contents.

#### *Extracting document properties as metadata*

11. Word document properties can be extracted as metadata. By default, only the Title will be extracted. Other properties can be extracted using the *extracted\_word\_metadata\_fields* option.
12. In the **Enrich** panel, look at the metadata that has been extracted for word05.doc and word06.doc. Now open the documents in Word and look at what properties they have set. (File --> Properties). They have **Title**, **Author**, **Subject**, and **Keywords** properties. WordPlug can be configured to look for these properties and extract them.
13. In the **Design** panel, under **Document Plugins**, select **WordPlug** and click **&ltConfigure Plugin>**. Switch on the configuration option **extracted\_word\_metadata\_fields**. Set the value to

```
Title,Author<Creator>,Subject,Keywords<Subject>
```

This will make WordPlug try to extract Title, Author, Subject and Keywords metadata. Title and Subject will be saved with the same name, while Author will be saved as Creator metadata, and Keywords as Subject metadata.

14. **Build** the collection.
15. Look at the metadata for the two documents again in the **Enrich** panel. You should now see these extra metadata items. This metadata can now be used in display or browsing classifiers etc.

## **Part V – PDF and PowerPoint documents**

---

Greenstone converts PDF and PowerPoint files to HTML using third-party software: pdftohtml.pl and ppttohtml.pl. This lets users view these documents even if they don't have the PDF or PowerPoint software installed. Unfortunately, sometimes the formatting of the resulting HTML files is not so good.

For these two document types, there is a new option for conversion: to a series of page/slide images. For PDF documents, Greenstone uses the ImageMagick convert program; for PowerPoint, it uses Windows native scripting (and therefore can only be applied on a Windows machine). Behind the scenes, the files are converted to a series of images with a corresponding item file, and processed using PagedImgPlug.

1. In the Librarian Interface, open up the **reports** collection you created previously. Add some PowerPoint documents to it, from **workshop-files→sample\_ppt**.
2. **Build** and **preview** the collection, and view the documents. Remember from **Lab 2** that pdf05-notext couldn't be processed during building, because there was no extracted text, and therefore doesn't appear in the collection. Note that the other PDF and PowerPoint documents appear as one long document, with no sections. The images are missing from the PowerPoint display.

### *Tidying up the HTML format for PDF documents*

3. In the **Design** panel, configure **PDFPlug**. Switch on the **use\_sections** option.  
  
**Build** and **preview** the collection. Note that all the PDF documents are now split into a series of pages, and a goto page box is provided. The format is still a bit ugly though.
4. In the **Design** panel, configure **PDFPlug**. Switch on the **complex** option. This will make PDFPlug use Ghostscript to try and generate nicer HTML. Ghostscript needs to be installed for this to work.

**Build** and **preview** the collection, and see how the format has changed to more closely resemble the original. In particular, you can see that pdf01.pdf has retained its columns in the HTML.

The PDF document with no text (pdf05-notext.pdf) now appears in the collection, but has no contents. The PDF with weird characters (pdf06-weirdchars.pdf) still does not display properly.

### *Using image format*

5. If conversion to HTML doesn't produce the result you like, both PDF and PowerPoint documents can be converted to a series of images, one per page or slide. For PDF, this requires ImageMagick to be installed, which is done as part of Greenstone installation from CD-ROM. For PowerPoint, this requires the use of Windows native scripting.
6. In the **Design** panel, configure **PDFPlug**. Set the **convert\_to** option to one of the image types, e.g. *pagedimg\_jpg*. Switch off the **use\_sections** and **complex** options, as they are not used with image conversion.
7. Configure **PPTPlug**. Switch on the **windows\_scripting** option, and change the **convert\_to** type to one of the image types, e.g. *pagedimg\_jpg*.
8. **Build** the collection and **preview**. All PowerPoint documents have been divided into sections, but only the extracted text is displayed. All PDF documents have been processed and divided into sections, but each section displays "This document has no text". For the conversion to images for PDF documents, no text is extracted.
9. In order to view the documents properly, you will need to modify the format statement. In the **Format Features** section on the **Design** panel, select the **DocumentText** format statement. Replace

[Text]

with

```
{If}{[parent:FileFormat] eq PDF,[srcicon],  
{If}{[parent:FileFormat] eq PPT,[srcicon],[Text]}}
```

Because the other documents in the collection do not use images, we only want to show images for PDF and PowerPoint documents. FileFormat is an extracted metadata item which shows the format of the source document. We use this to test whether the documents are PDF/PPT or not.

10. **Preview** the collection from the **Create** panel. (There is no need to build it). Images from the document are now displayed instead of the extracted text. Both pdf05-

notext.pdf and pdf06-weirdchars.pdf display nicely now. Make sure that the word documents still display properly.

*Using process\_exp to control document processing (Optional, very advanced)*

11. Processing all of the PDF documents using an image type may not give the best result for your collection. The images will look nice, but as no text is extracted, searching the full text will not be available for these documents. The best solution would be to process most of the PDF files as HTML, and only use the image format where HTML doesn't work.
12. We achieve this by adding two PDFPlug plugins to the collection, with different options. Currently, the GLI does not allow you to add the same plugin twice to the collection (with the exception of UnknownPlug). You will need to edit the collection configuration file by hand. Close the reports collection in GLI. Then open **C:\Program Files\Greenstone\collect\reports\etc\collect.cfg** using WordPad. In the list of plugins, add another PDFPlug, i.e.

```
plugin PDFPlug
```

Don't worry about the options here – we will add these using the GLI.

Note that if you ever need to edit a collection's collect.cfg file by hand, you must close the collection in GLI first, otherwise the next time GLI saves the file, it will overwrite your changes.

13. Open up the collection again in the Librarian Interface, and go to the **Gather** panel. Make a new folder called "notext". Right click in the collection panel and select **New folder** from the menu. Change the **Folder Name** to notext, and click <OK>. Move the two pdf files that have problems with html (pdf05-notext.pdf and pdf06-weirdchars.pdf) into this folder by drag and drop. We will set up the plugins so that PDF files in this "notext" directory are processed differently to the other PDF files.
14. Switch to the **Document Plugins** section of the **Design** panel. You will see that there are two **PDFPlug** plugins in the list.
15. Switch to **Library Systems Specialist** mode, as you will need to use regular expressions in the options.
16. Configure the two **PDFPlug** plugins so that the options look like the following:

```
plugin PDFPlug -convert_to pagedimg_jpg -process_exp "notext.*\.pdf"  
plugin PDFPlug -convert_to html -use_sections
```

The paged\_img version must come earlier in the list than the html version. The process\_exp for the first **PDFPlug** will process any PDF files in the notext directory.

The second **PDFPlug** will process any PDF files that are not processed by the first one.

Note that all plugins have the `process_exp` option, and this can be used to customize which documents are processed by which plugin. This option is only visible in **Library Systems Specialist** and **Expert** modes.

Change back to **Librarian** mode.

17. Edit the **DocumentText** format statement. PDF files processed as HTML will not have images to display, so we need to make sure they get text displayed instead.

Change the first `[srcicon]` element in the following part with

`{Or}{[srcicon],[Text]}`. I.e. change

```
{If}{[parent:FileFormat] eq PDF,[srcicon],  
{If}{[parent:FileFormat] eq PPT,[srcicon],[Text]}}
```

to

```
{If}{[parent:FileFormat] eq PDF, {Or}{[srcicon],[Text]},  
{If}{[parent:FileFormat] eq PPT,[srcicon],[Text]}}
```

18. **Build** and **preview** the collection. All PDF and PowerPoint documents should look relatively nice. Try searching this collection. You will be able to locate the PDFs that were converted to HTML (try e.g. “bibliography”), but not the ones that were converted to images (try searching for “banana” or “METS”).

## Part VI – CDS/ISIS

---

1. Create a new collection in the Librarian Interface called **ISIS Collection**. Add the files from *workshop\_files/sample\_isis* into the collection. Click **<OK>** to add **ISISPlug** when prompted.
2. **Build** and **preview** the collection. The default indexes, classifiers and document display are not very useful for this data.
3. Follow the instructions given in Lecture 6 to customize the collection.
4. Improve the collection further by indexing, classifying and displaying another metadata element of your choice.

## Part VII – Your custom collection

---

Using any of the advanced features you have worked on in this lab, customize your own collection.