Lab 3: Advanced collection configuration

3.1. Formatting the Word and PDF collection

In this exercise, we play around with the format statements in the Word and PDF collection.

1. Open the **reports** collection in the Librarian Interface and go to the **Format Features** section of the **Format** panel.

Tidying up the default format statement

2. In this part of the exercise, we make the format statement simpler without changing the resulting display.

   Greenstone's default format statement is complex because it is designed to produce something reasonable under almost any conditions, and also because for practical reasons it needs to be backwards compatible with legacy collections. For this collection, we don't need all of the complexity.

   Make sure that the **VList** format statement is selected in the list of formats.

   The default **VList** format statement looks like the following:

   ```
   <td valign="top">[link][icon][/link]</td>
   <td valign="top">[ex.srclink]{Or}{[ex.thumbicon],[ex.srcicon]}[ex./srclink]</td>
   <td valign="top">[highlight]
   {Or}{[dc.Title],[exp.Title],[ex.Title],Untitled}
   [/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}</td>
   ```

   This format statement is the default used for any vertical list, such as search results, classifiers, and document table of contents.

   `{Or}{[ex.thumbicon],[ex.srcicon]}` chooses *ex.thumbicon* metadata if its there, otherwise chooses *ex.srcicon* metadata. If neither are present, nothing is displayed. For this collection there is no *ex.thumbicon* metadata so the choice is not needed.

   Replace `{Or}{[ex.thumbicon],[ex.srcicon]}` with `[ex.srcicon]`.

   There is no *exp.Title* metadata, so remove that element from `{Or}{[dc.Title],[exp.Title],[ex.Title],Untitled}`.

   The resulting format statement looks like the following:

   ```
   <td valign=top>[link][icon][/link]</td>
   <td valign=top>[ex.srclink][ex.srcicon][ex./srclink]</td>
   <td valign=top>[highlight]
   {Or}{[dc.Title],[ex.Title],Untitled}[/highlight] {If}{[ex.Source],<br><i>([ex.Source])</i>}</td>
   ```

   Preview the collection to make sure the display hasn't changed. You shouldn't notice any difference when looking at search results, classifiers etc.

Linking to Greenstone version or original version of documents

3. For collections with documents that undergo a conversion process during importing (e.g. Word, PDF, PowerPoint documents, but not text, HTML documents), the original file is stored in the collection along with the converted version. The default **VList** format statement links to both versions:

   `[link][icon][/link]` links to the Greenstone HTML version, while `[ex.srclink][ex.srcicon][/ex.srclink]` links to the original.

   Choose **SearchVList** in **Format Features** by selecting **Search** from the **Choose Feature** drop down list, and **VList** from the **Affected Component** list. Click **<Add Format>** to add the **SearchVList** format statement into the list of assigned formats. Experiment with removing either of the two links from the format statement.

To see the results of your changes, preview the collection and do a search. You are making changes to **SearchVList**, which means the changes will only apply to search results.

Storing and displaying the original allows users to see the correct format, but requires the user to have the relevant program installed. It also increases the size of the collection. The Greenstone version can be viewed in a browser, but may not look as nice.

Making bookshelves show how many items they contain

4. Next, we'll customize the format for the *Creators* list. Classifier bookshelves have only a few pieces of metadata to display: `[ex.Title]` and `[numleafdocs]`. Whatever metadata the classifier has been built on, the bookshelf label is always stored as `[ex.Title]`. This is why a Creator is printed out for each bookshelf even though `[dc.Creator]` is not specified in the format statement. `[numleafdocs]` is only defined for bookshelves, so this metadata can be used in an `{If}` statement to make bookshelves and documents display differently in the list.

   Make each bookshelf in the Creator classifier show how many entries it contains. In the **Format Features** section of the **Format** panel, select the **CL2 AZCompactList** classifier which is based on **dc.Creator** metadata from the **Choose Feature** drop down list, and **VList** from the **Affected Component** list. Click the **<Add Format>** button to add this format into the list of assigned formats. Note that it gets added as **CL2VList** in this list: it is the **VList** format for the second (**CL2**) classifier.

   Append the following text to the bottom of the format statement:

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

   Preview the collection. Click on the *Creators* list and notice that the bookshelves now display how many documents they contain.

   This revised format statement has the effect of specifying in brackets how many items are contained within a bookshelf. Since only bookshelves define `[numleafdocs]`, only they will display this. By modifying **CL2VList** instead of **VList**, the change will only apply to the second classifier (Creators).

Displaying multi-valued metadata

5. Next we modify the document entries in the Creator classifier to display all authors. Back in **Format Features**, select the **CL2VList** format in the list of assigned formats. After `{If}{[ex.Source],<br>` in the format statement, add `[sibling:dc.Creator]`.

   `[ex.Source]` is not defined for bookshelves, so can also be used to differentiate bookshelves and documents.

   The resulting format statement looks like:

   ```
   <td valign=top>[link][icon][/link]</td>
   <td valign=top>[ex.srclink][ex.srcicon][ex./srclink]</td>
   <td valign=top>[highlight]
   {Or}{[dc.Title],[ex.Title],Untitled}[/highlight]
   {If}{[ex.Source],<br>[sibling:dc.Creator]
   <i>([ex.Source])</i>}</td>
   {If}{[numleafdocs],<td><i>([numleafdocs])</i></td>}
   ```

   This will display the Greenstone link, the link to the original, then the Title. For bookshelves, it will also display how many documents the bookshelf contains. For documents, it will display all the Authors (Creators), and the source document. `[sibling:dc.Creator]` displays all the Creator metadata for the document, separated by a space (" "), while `[dc.Creator]` displays only the first author. Preview the *Creators* list and make sure that all authors are displayed for documents.

6. You can change the separator between the authors. Modify the format statement, and replace `[sibling:dc.Creator]` with `[sibling(All'<br/>'):dc.Creator]`. This will add a new line after each author (`<br/>` specifies a line break in HTML). Preview the *Creators* list.

   If you have done exercise **Enhanced Word document handling**, the collection will have both dc.Creator and ex.Creator metadata. To display both, you can use

   ```
   [sibling:dc.Creator] [sibling:ex.Creator]
   ```

   To display dc.Creator if it is present, otherwise display ex.Creator, use

   ```
   {Or}{[sibling:dc.Creator],[sibling:ex.Creator]}
   ```

Opening PDF files with query terms highlighted

7. Next we'll customize the **SearchVList** format statement to highlight the query terms in a PDF file when it is opened from the search result list. This requires Acrobat Reader 7.0 version or higher, and currently only works on a Microsoft Windows platform.

8. The search terms are kept in the macro variable **_cgiargq_**, and we append **#search="_cgiargq_"** to the end of a PDF file link to pass the query terms to the PDF file.

   PDFPlug renames each PDF file as **doc.pdf** and saves it in a unique directory for that document, so we use

   ```
   _httpcollection_/index/assoc/[archivedir]/doc.pdf
   ```

   to refer to the PDF source file. (However, if you used the **-keep_original_filename** option to **PDFPlug** when building the collection, the original name of the PDF file is kept, and we use

   ```
   _httpcollection_/index/assoc/[archivedir]/[Source]
   ```

   instead to locate the PDF source file.)

9. Select **SearchVList** from the list of assigned formats. We need to test whether the file is a PDF file before linking to doc.pdf, using `{If}{[ex.FileFormat] eq 'PDF',,}`. For PDF files, we use the above format instead of the `[ex.srclink]` and `[ex./srclink]` variables to link to the file.

   The resulting format statement is:

   ```
   <td valign="top">[link][icon][/link]</td>
   <td valign="top">{If}{[ex.FileFormat] eq 'PDF', <a
   href=\"_httpcollection_/index/assoc/[archivedir]/doc.pdf#search=&quot;_cgiargq_&quot;\">[ex.srcicon]</a>,
   [ex.srclink][ex.srcicon][ex./srclink]}</td>
   <td valign="top">[highlight]
   {Or}{[dc.Title],[ex.Title],Untitled}
   [/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}</td>
   ```

   When the PDF icons are clicked in the search results, Acrobat will open the file with the search window open, and the query terms highlighted.

3.2. Formatting the HTML collection—Tudor

1. Open up your **tudor** collection, go to the **Format** panel (by clicking on its tab) and select **Format Features** from the left-hand list. Leave the editing controls at their default value, so that **Choose Feature** displays *All Features* and **VList** is selected as the **Affected Component**. The text in the **HTML Format String** box reads as follows:

   ```
   <td valign=top>[link][icon][/link]</td>
   <td valign=top>[ex.srclink]{Or}{[ex.thumbicon],[ex.srcicon]} [ex./srclink]</td>
   <td valign=top>[highlight]
   {Or}{[dc.Title],[exp.Title],[ex.Title],Untitled}
   [/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}</td>
   ```

   This displays something that looks like this:

   > A discussion of question five from Tudor Quiz: Henry VIII
   > *(quizstuff.html)*

   for a particular document whose *Title* metadata is **A discussion of question five from Tudor Quiz: Henry VIII** and whose *Source* metadata is **quizstuff.html**.

   This format appears in the search results list, in the **Titles** list, and also when you get down to individual documents in the **Subjects** hierarchy. This is Greenstone's default format statement.

Greenstone's default format statement is complex because it is designed to produce something reasonable under almost any conditions, and also because for practical reasons it needs to be backwards compatible with legacy collections.

2. Delete the contents of the **HTML Format String** box and replace it with this simpler version:

   ```
   <td>[link][icon][/link]</td>
   <td>[ex.Title]<br>
       <i>([ex.Source])</i>
   </td>
   ```

   Preview the result (you don't need to build the collection, because changes to format statements take effect immediately). Look at some search results and at the **Titles** list. They are just the same as before! Under most circumstances this far simpler format statement is entirely equivalent to Greenstone's more complex default.

   But there's a problem. Beside the bookshelves in the ***Subjects*** *browser, beneath the subject appears a mysterious "()". What is printed for these bookshelves is governed by the same format statement, and though bookshelf nodes of the hierarchy have associated* Title *metadata—their title is the name of the metadata value associated with that bookshelf—they do not have* ***ex.Source*** *metadata, so it comes out blank.*

3. In the **Format Features** section of the **Format** panel, the **Choose Feature** menu (just above **Affected Component** menu) displays *All Features*. That implies that the same format is used for the search results, titles, and all nodes in the subject hierarchy—including internal nodes (that is, bookshelves). The **Choose Feature** menu can be used to restrict a format statement to a specific one of these lists. We will override this format statement for the hierarchical *subject* classifier. In the **Choose Feature** menu, scroll down to the item that says

   > CL2: Hierarchy -metadata dc.Subject and Keywords

   and select it. This is the format statement that affects the second classifier (i.e., "CL2"), which is a **Hierarchy** classifier based on **dc.Subject and Keywords** metadata.

   Click **<Add Format>** to add this format statement to the collection.

   Edit the **HTML Format String** box below to read

   ```
   <td>[link][icon][/link]</td>
   <td>[ex.Title]</td>
   ```

4. **Preview** the **Subjects** list in the collection. First, the offending "()" has disappeared from the bookshelves. Second, when you get down to a list of documents in the subject hierarchy, the filename does not appear beside the title, because **ex.Source** is not specified in the format statement and this format statement applies to all nodes in the *subject* classifier. Note that the search results and titles lists have not changed: they still display the filename underneath the title.

5. Let's change the search results format so that **dc.Subject and Keywords** metadata is displayed here instead of the filename. In the **Choose Feature** menu (under **Format Features** on the **Format** panel), scroll down to the item **Search** and select it. Click **<Add Format>** to add this format statement to the collection. Change the **HTML Format String** box below to read

```
<td>[link][icon][/link]</td>
<td>[ex.Title]<br>
    [dc.Subject]
</td>
```

6. To insert the **[dc.Subject]**, position the cursor at the appropriate point and either type it in, or select it from the **Insert Variable...** drop down menu. This menu shows many of the things that you can put in square brackets in the format statement.

7. **Preview** the collection. Documents in the search results list will be displayed like this:

   A discussion of question five from Tudor Quiz: Henry VIII
   Tudor period|Others

   (The vertical bar appears because this **dc.Subject and Keywords** metadata is hierarchical metadata. Unfortunately there is no way to get at individual components of the hierarchy. For most metadata, such as title and author, this isn't a problem.)

8. Finally, let's return to the *Subjects* hierarchy and learn how to do different things to the bookshelves and to the documents themselves. In the **Choose Feature** menu, re-select the item

   CL2: Hierarchy -metadata dc.Subject and Keywords

   Edit the **HTML Format String** box below to read

```
<td>[link][icon][/link]</td>
<td>{If}{[numleafdocs],<b>Bookshelf title:</b> [ex.Title],
         <b>Title:</b> [ex.Title]}
</td>
```

   Again, you can insert the items in square brackets by selecting them from the **Insert Variable...** drop down box.

   *The **If** statement tests the value of the variable **numleafdocs**. This variable is only set for internal nodes of the hierarchy, i.e. bookshelves, and gives the number of documents below that node. If it is set we take the first branch, otherwise we take the second. Commas are used to separate the branches. The curly brackets serve to indicate that the **If** is special—otherwise the word "If" itself would be output.*

9. **Preview** the collection and examine the subject hierarchy again to see the effect of your changes. Bookshelves should say **Bookshelf title:** and then the title, while documents will display **Title:** and the title. Note that the number of documents in the bookshelf is not displayed: we are using `[numleafdocs]` to test what kind of item in the list we are at, but we are not displaying it.

3.3. Pointing to documents on the web

1. Open up your **webtudor** collection, and in the **Gather** panel inspect the files you dragged into it. The first folder is *englishhistory.net*, which opens up to reveal *tudor*, and so on. The files represent a complete sweep of the pages (and supporting images) that constitute the *Tudor citizens* section of the *englishhistory.net* web site. They were downloaded from the web 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.

2. In the **Design** panel, select the **Document Plugins** section, then select the **plugin HTMLPlug** line and click **<Configure Plugin...>**. A popup window appears. Locate the **file_is_url** option (about halfway down the first block of items) and switch it on. While you are there, switch off the **smart_block** option so that stray images are not processed. Click **<OK>**.

   Setting this option to the **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, for example the *tudor* folder, because this will set **URL** metadata to something like

   > http://tudor/citizens/...

   rather than

   > http://englishhistory.net/tudor/citizens/...

   If you have copied over a subfolder previously, delete it and make a fresh copy. Drag the folder in the right-hand side of the **Gather** panel on to the trash can in the lower right corner. Then obtain a fresh copy of the files by dragging across the *englishhistory.net* folder from the **Downloaded Files** folder on the left-hand side.

3. 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. Go to the **Format** panel, select the **Format Features** section and edit the **VList** format statement by replacing

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

   with

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

4. 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 (assuming it is still there by the time you do this exercise!). If you are working offline you will be unable to retrieve the document.

3.4. Section tagging for HTML documents

1. In a browser, 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 a text editor (e.g. WordPad) open up one of the HTML files from the demo collection: *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>
-->
```

   When 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 browser, find the **Farming snails 1** document in the demo collection (through the *Titles* 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, lets add a new subsection into the **Introduction** chapter. In the text editor, add 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.

5. **Build** and **preview** the collection. Look at the **Farming snails 1** document again and check that your new section has been added.

3.5. Exporting a collection to CD-ROM/DVD

To publish a collection on CD-ROM or DVD, Greenstone's Export to CD-ROM export module must be installed. This is included with CD-ROM distributions, and all distributions 2.70w and later. It must be installed separately for non-CD-ROM versions of Greenstone, version 2.70 and earlier (see ***Installing Greenstone****).*

1. Launch the Greenstone Librarian Interface if it is not already running.

2. Choose **File** **Write CD/DVD image...**. In the resulting popup window, select the collection or collections that you wish to export by ticking their check boxes. You can optionally enter a name for the CD-ROM: this is the name that will appear in the menu when the CD-ROM is run. If a name is not entered, the default **Greenstone Collections** will be used. You can also specify whether the resulting CD-ROM will install files onto the host machine when used or not. Click **<Write CD/DVD image>** to start the export process.

   The necessary files for export are written to:

   Greenstone tmp exported_xxx

   where xxx will be similar to the name you have entered. If you didn't specify a name for the CD-ROM, then the folder name will be *exported_collections*.

   You need to use your own computer's software to write these on to CD-ROM. On *Windows XP* this ability is built into the operating system: assuming you have a CD-ROM or DVD writer insert a blank disk into the drive and drag the *contents* of *exported_xxx* into the folder that represents the disk.

   The result will be a self-installing Windows Greenstone CD-ROM or DVD, which starts the installation process as soon as it is placed in the drive.

3.6. Enhanced PDF handling

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

This exercise explores some extra options to the PDF plugin which may produce a nicer version for display. Some of these options use the standard pdftohtml program, others use ImageMagick and Ghostscript to convert the file to a series of images. Ghostscript is a program that can convert Postscript and PDF files to other formats. You can download it from http://www.cs.wisc.edu/~ghost/ (follow the link to the current stable release).

1. In the Librarian Interface, start a new collection called "PDF collection" and base it on **-- New Collection --**.

   In the **Gather** panel, drag just the PDF documents from *sample_files Word_and_PDF Documents* into the new collection. Also drag in the PDF documents from *sample_files Word_and_PDF difficult_pdf* .

   Go to the **Create** panel and build the collection. Examine the output from the build process. You will notice that one of the documents could not be processed. The following messages are shown: "The file pdf05-notext.pdf was recognised but could not be processed by any plugin.", and "3 were processed and included in the collection. 1 was rejected".

2. Preview the collection and view the documents. *pdf05-notext.pdf* does not appear as it could not be processed. *pdf06-weirdchars.pdf* was processed but looks very strange. The other PDF documents appear as one long document, with no sections.

Modes in the Librarian Interface

*The Librarian Interface can operate in different modes. The default mode is **Librarian** mode. We can use **Expert** mode to work out why the pdf file could not be processed.*

3. Use the **Preferences...** item on the **File** menu to switch to **Expert** mode and then build the collection again. The **Create** panel looks different in **Expert** mode because it gives more options: locate the **<Build Collection>** button, near the bottom of the window, and click it. Now a message appears saying that the file could not be processed, and why. Amongst all the output, we get the following message: "Error: PDF contains no extractable text. Could not convert pdf05-notext.pdf to HTML format". pdftohtml.pl cannot convert a PDF file to HTML if the PDF file has no extractable text.

4. We recommend that you switch back to **Librarian** mode for subsequent exercises, to avoid confusion.

Splitting PDFs into sections

5. In the **Document Plugins** section of the **Design** panel, configure **PDFPlug**. Switch on the **use_sections** option.

   In the **Search Indexes** section, check the **section** checkbox to build the indexes on section level as well as document level.

   Build and **preview** the collection. View the text versions of some of the PDF documents. Note that these are now split into a series of pages, and a "go to page" box is provided. The format is still a bit ugly though, and pdf05-notext.pdf is still not processed.

Using image format

6. If conversion to HTML doesn't produce the result you like, PDF documents can be converted to a series of images, one per page. This requires ImageMagick and Ghostscript to be installed.

7. In the **Document Plugins** section, configure **PDFPlug**. Set the **convert_to** option to one of the image types, e.g. **pagedimg_jpg**. Switch off the **use_sections** option, as it is not used with image conversion.

8. **Build** the collection and **preview**. All PDF documents (including pdf05-notext.pdf) 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 **Format** panel, select the **DocumentText** format statement. Replace

   ```
   [Text]
   ```

   with

   ```
   [srcicon]
   ```

10. Preview the collection. Images from the document are now displayed instead of the extracted text. Both *pdf05-notext.pdf* and *pdf06-weirdchars.pdf* display nicely now.

    In this collection, we only have PDF documents and they have all been converted to images. If we had other document types in the collection, we should use a different format statement, such as:

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

    ***FileFormat** is an extracted metadata item which shows the format of the source document. We can use this to test whether the documents are PDF or not: for PDF documents, display [srcicon], for other documents, display [Text].*

Using process_exp to control document processing (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 putting the problem files into a separate folder, and adding another **PDFPlug** plugin with different options.

13. 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* folder are processed differently to the other PDF files.

14. Change to **Library Systems Specialist** mode so that you can add two of the same plugin, and use regular expressions in the plugin options (**File** **Preferences...** **Mode**).

    For version 2.71, you'll need to close GLI now then restart it to get the list of plugins to update properly.

15. Switch to the **Document Plugins** section of the **Design** panel. Add a second PDF plugin by selecting **PDFPlug** from the **Select plugin to add:** drop-down list, and clicking **<Add Plugin...>**. This plugin will come after the first PDF plugin, so we configure it to process PDF documents as HTML. Set the **convert_to** option to **html**, and switch on the **use_sections** option. Click **<OK>**.

16. Configure the first PDF plugin, and set the **process_exp** option to **'notext.*\.pdf'**.

17. The two PDF plugins should have options 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.

18. 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 `[srcicon]` to `{If}{[NoText] eq "1",[srcicon],[Text]}`.

19. Build and preview the collection. All PDF documents should look relatively nice. Try searching this collection. You will be able to search for the PDFs that were converted to HTML (try e.g. "bibliography"), but not the ones that were converted to images (try searching for "FAO" or "METS").

3.7. Bibliographic collection—Part B

Exploding the database

16. Go to the **Enrich** panel and try to see the metadata. It doesn't appear! This is because the metadata is associated with records inside the file, not the file itself.

    Metadata file types, such as MARC, CDS/ISIS, BibTex etc. can be imported into Greenstone but their metadata cannot be viewed in the Librarian Interface. To edit any metadata you need to go back to the program that created the file.

    Greenstone provides a way of *exploding* a metadata database so that each record appears as an individual document, with viewable and editable metadata. This process is irreversible: once this step has been done, the database is deleted and can no longer be used in its original program.

17. In the **Gather** panel, you may notice that the MARC database has a different coloured icon to other files. This green icon indicates that a file is a metadata database that can be exploded. Right-click on the file and choose **Explode Metadata Database** from the menu. A new window opens, containing options for the exploding process. A description of each option can be obtained by hovering the mouse over the option.

    Turn on the **metadata_set** option by checking its box. This option indicates which metadata set to explode the metadata into. The default set is the "Exploded Metadata Set"—a metadata set which initially has no elements in it, but will receive a new element for each metadata field retrieved from the database.

18. Click **<Explode>** to start the exploding process. This may take a short while, depending on the size of the database.

19. Once exploding has finished, the MARC database file will have been deleted, and three folders created in its place. These folders contain an empty file for each record in the original database. The metadata for these records can be viewed and edited by switching to the **Enrich** panel.

20. Because the MARC file is no longer present, and the collection contains empty (.nul) files, we need to change the list of plugins. In the **Document Plugins** section of the **Design** panel, remove **MARCPlug** and add **NULPlug** (use the default configuration).

21. **Rebuild** and **preview** the collection. You will notice that the *Subjects* classifier is empty, searching no longer returns any results, and the document display is useless.

    Although the *Titles* classifier was built on **ex.Title**, it still displays the correct titles, but in the **Enrich** panel you can see the **ex.Title** metadata are actually the filenames rather than titles of the MARC records. This is because the default **VList** format uses the **exp.Title** metadata. In the **Format Features** section of the **Format** panel, select **VList** in the list of assigned format statements. The resulting format statement looks like:

    ```
    <td valign="top">[link][icon][/link]</td>
    <td valign="top">[ex.srclink]{Or}{[ex.thumbicon],[ex.srcicon]}[ex./srclink]</td>
    <td valign="top">[highlight]
    {Or}{[dc.Title],[exp.Title],[ex.Title],Untitled}
    [/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}
    ```

    Since there is no **dc.Title** metadata and **exp.Title** comes before **ex.Title**, the expoded titles will be displayed.

Reformatting the collection to use the exploded metadata

The collection previously used extracted (ex.) metadata, but now it uses exploded (exp.) metadata. The *Subjects* classifier and search indexes were built on ex metadata, which is why they no longer work properly.

There is also no longer any text in the documents. Previously, MARCPlug stored the raw record as the "text" of each record. Now that the metadata is in the Librarian Interface, there is no longer the concept of raw record, and so there is no text.

We need to modify the collection design to take note of these changes.

22. In the **Search Indexes** section, change the Title index to use **exp.Title**: select the Title index in the **Assigned Indexes** list and click **<Edit Index>**. Deselect **ex.Title** in the list of metadata, and select **exp.Title**. Click **<Replace Index>**.

23. Remove the **ex.Subject** index by selecting it in the **Assigned Indexes** list and clicking **<Remove Index>**. Add an index on **exp.Subject**: click **<New Index>**, select **exp.Subject** in the metadata list, and click **<Add Index>**.

24. The text index is no longer any use, so remove that index too.

25. To enable combined searching across all indexes at once, click **<New Index>**, tick the **Add combined searching over all assigned indexes (allfields)** checkbox, and click **<Add Index>**. Move this to the top of the list using the **<Move Up>** button, so that it appears first in the drop down list. Click **<Set Default Index>** so that it becomes the default field for searching.

26. To explicitly use the **exp.Title** metadata, in the **Browsing Classifiers** section, change the **ex.Title AZList** to use **exp.Title** metadata. Double click the **ex.Title AZList** in the **Assigned Classifiers** list, and change the **metadata** option to use **exp.Title**. Click **<OK>**. Do the same thing for the Subject **AZCompactList**, changing **ex.Subject** to **exp.Subject**.

27. In the **Format Features** section of the **Format** panel, select **VList** in the list of assigned format statements.

    - There is no dc metadata for this collection, so replace `{Or}{[dc.Title],[exp.Title],[ex.Title],Untitled}` with `{Or}{[exp.Title],[ex.Title],Untitled}`.
    - There are no source or thumb icons, so remove the second line: `<td valign="top">[ex.srclink]{Or}{[ex.thumbicon],[ex.srcicon]}[ex./srclink]</td>`.
    - The ex.Source metadata is set to the nul filename, so remove that from the display: remove `{If}{[ex.Source],<br><i>([ex.Source])</i>}`

    The resulting format statement looks like:

    ```
    <td valign="top">[link][icon][/link]</td>
    <td valign="top">[highlight]
    {Or}{[exp.Title],[ex.Title],Untitled}
    [/highlight]</td>
    ```

28. Clear the **DocumentHeading** format statement by selecting it in the list of assigned format statements and deleting the contents in the **HTML Format String**. The record Title will be displayed as part of the **DocumentText** format, so we don't need it here.

29. Next, edit the **DocumentText** format statement. Delete the contents and replace it with

    ```
    <table>
    <tr><td>Title:</td><td>[exp.Title]</td></tr>
    <tr><td>Subject:</td><td>[exp.Subject]</td></tr>
    <tr><td>Publisher:</td><td>[exp.Publisher]</td></tr>
    </table>
    ```

30. The *DETACH* and *NO HIGHLIGHTING* buttons are not very useful for this collection, so lets get rid of them. Edit the **DocumentButtons** format statement to make it empty.

31. **Rebuild** and **preview** the collection. The classifiers should be back to normal, searching should now work, and there should be a nice record display.