Lab 7: Two examples—multimedia and scanned images

7.1. Looking at a multimedia collection

1. Copy the entire folder

   sample_files → *beatles* → *advbeat_large*

   (with all its contents) into your Greenstone *collect* folder. If you have installed Greenstone in the usual place, this is

   My Computer → *Local Disk (C:)* → *Program Files* → *Greenstone* → *collect*

   Put *advbeat_large* in there.

2. If the Greenstone Digital Library Local Library Server is already running, re-start it by clicking the CD icon on the task bar and then pressing *Restart Library*. If not, start it up by selecting *Greenstone Digital Library* from the *Start* menu.

3. Explore the Beatles collection. Note how the *Browse* button divides the material into seven different types. Within each category, the documents have appropriate icons. Some documents have an audio icon: when you click these you hear the music (assuming your computer is set up with appropriate player software). Others have an image thumbnail: when you click these you see the images.

4. Look at the *Titles* browser. Each title has a bookshelf that may include several related items. For example, *Hey Jude* has a MIDI file, lyrics, and a discography item.

5. Observe the low quality of the metadata. For example, the four items under **A Hard Day's Night** (under "H" in the *Titles* browser) have different variants as their titles. The collection would have been easier to organize had the metadata been cleaned up manually first, but that would be a big job. Only a tiny amount of metadata was added by hand—fewer than ten items. The original metadata was left untouched and Greenstone facilities used to clean it up automatically. (You will find in **Building a multimedia collection** that this is possible but tricky.)

6. In the Windows file browser, take a look at the files that makes up the collection, in the

   sample_files → *beatles* → *advbeat_large* → *import*

   folder. What a mess! There are over 450 files under seven top-level sub-folders. Organization is minimal, reflecting the different times and ways the files were gathered. For example, *html_lyrics* and *discography* are excerpts of web sites, and *images* contains various images in JPEG format. For each type, drill down through the hierarchy and look at a sample document.

7.2. Building a multimedia collection

We will proceed to reconstruct from scratch the Beatles collection that you have just looked at. We develop the collection using a small subset of the material, purely to speed up the repeated rebuilding that is involved.

1. Start a new collection (**File** → **New...**) called **small beatles**, basing it on the default **-- New Collection --**. (Basing it on the existing Advanced Beatles collection would make your life far easier, but we want you to learn how to build it from scratch!)

2. Copy the files provided in

   sample_files → *beatles* → *advbeat_small*

   into your new collection. Do this by opening up *advbeat_small*, selecting the eight items within it (from *discography* to *beatles_midi.zip*), and dragging them across. Because some of these files are in MP3 and MARC formats you will be asked whether to include **MP3Plug** and **MARCPlug** in your collection. Click **<Add Plugin>**.

3. Change to the **Enrich** panel and browse around the files. There is no metadata—yet. Recall that you can double-click files to view them.

   (There are no MIDI files in the collection: these require more advanced customisation because there is no MIDI plugin. We will deal with them later.)

4. Change to the **Create** panel and **build** the collection.

5. **Preview** the result.

Manually correcting metadata

6. You might want to correct some of the metadata—for example, the atrocious misspelling in the titles "MAGICAL MISTERY TOUR." These documents are in the discography section, with filenames that contain the same misspelling. Locate one of them in the **Enrich** panel. Notice that the extracted metadata element **ex.Title** is now filled in, and misspelt. You cannot correct this element, for it is extracted from the file and will be re-extracted every time the collection is re-built.

7. Instead, add **dc.Title** metadata for these two files: "Magical Mystery Tour." Change to the **Enrich** panel, open the discography folder and drill down to the individual files. Set the **dc.Title** value for the two offending items.

*Now there's a twist. The **dc.Title** metadata won't appear in Titles because the classifier has been instructed to use **ex.Title**. But changing the classifier to use **dc.Title** would miss out all the extracted titles! Fortunately, there's a way of dealing with this by specifying a list of metadata names in the classifier.*

8. Change to the **Design** panel and select the **Browsing Classifiers** section. Double-click the **ex.**

Title classifier (the first one) to edit its configuration settings.

- Type `dc.Title,` before the `ex.Title` in the metadata box—i.e. make it read

  ```
  dc.Title,ex.Title
  ```

  and click **<OK>**.

Build the collection again, and **preview** it.

Extracted metadata is unreliable. But it is very cheap! On the other hand, manually assigned metadata is reliable, but expensive. The previous section of this exercise has shown how to aim for the best of both worlds by using extracted metadata but correcting it when it is wrong.

Browsing by media type

9. First let's remove the **AZList** classifier for filenames, which isn't very useful, and replace it with a browsing structure that groups documents by category (discography, lyrics, audio etc.). Categories are defined by manually assigned metadata.

   - Change to the **Enrich** panel, select the folder *discography* and set its **dc.Format** metadata value to "Discography". Setting this value at the folder level means that all files within the folder inherit it.
   - Repeat the process. Assign "Lyrics" to the *html_lyrics* folder, "Images" to *images*, "MARC" to *marc*, "Audio" to *mp3*, "Tablature" to *tablature_txt*, and "Supplementary" to *wordpdf*.
   - Switch to the **Design** panel and select the **Browsing Classifiers** section.
   - Delete the **ex.Source** classifier (the second one).
   - Add an **AZCompactList** classifier. Select **dc.Format** as the **metadata** field and specify "browse" as the **buttonname**. Click the **sort** checkbox, and select **ex.Title** in the drop-down list: this will make the classifier display documents in alphabetical order of title.

   Build the collection again and **preview** it.

*Note how we assigned **dc.Format** metadata to all documents in the collection with a minimum of labour. We did this by capitalizing on the folder structure of the original information. Even though we complained earlier about how messy this folder structure is, you can still take advantage of it when assigning metadata.*

Suppressing dummy text

10. Alongside the Audio files there is an MP3 icon, which plays the audio when you click it, and also a text document that contains some dummy text. Image files also have dummy documents. These dummy documents aren't supposed to be seen, but to suppress them you have to fiddle with a format statement.

    - Change to the **Format** panel and select the **Format Features** section.
    - Ensure that **VList** is selected, and make the changes that are highlighted below. You need to insert five lines into the first line, and delete the second line. (Note, the changes are available in a text file, see below.) Change:

```
<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}{[dls.Title],[dc.Title],[Title],Untitled}
[/highlight]{If}{[ex.Source],<br><i>([ex.Source])</i>}</td>
```

to this:

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

To make this easier for you we have prepared a plain text file that contains the new text. In WordPad open the following file:

sample_files → *beatles* → *format_tweaks* → *audio_tweak.txt*

(Be sure to use WordPad rather than Notepad, because Notepad does not display the line breaks correctly.) Place it in the copy buffer by highlighting the text in WordPad and selecting **Edit** → **Copy**. Now move back to the Librarian Interface, highlight all the text that makes up the current **VList** format statement, and use **Edit** → **Paste (ctrl-v)** to transform the old statement to the new one.

Preview the result. You may need to click the browser's <**Reload**> button to force it to re-load the page.

11. While we're at it, let's remove the source filename from where it appears after each document.

    - In the **VList** format feature, delete the text that is highlighted below:

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

Preview the result (you don't need to rebuild the collection.)

Using AZCompactList rather than AZList

12. There are sometimes several documents with the same title. For example, *All My Loving* appears both as lyrics and tablature (under *ALL MY LOVING*). The **Titles** browser might be improved by

grouping these together under a bookshelf icon. This is a job for an **AZCompactList**.

- Change to the **Design** panel and select the **Browsing Classifiers** section.
- Remove the **ex.Title** classifier (at the top)
- Add an **AZCompactList** classifier, and enter **dc.Title,ex.Title** as its metadata.
- Finish by pressing **<OK>**.
- Move the new classifier to the top of the list (**<Move Up>** button).

Build the collection again and **preview** it. Both items for *All My Loving* now appear under the same bookshelf. However, many entries haven't been amalgamated because of non-uniform titles: for example *A Hard Day's Night* appears as four different variants. We will learn below how to amalgamate these.

Making bookshelves show how many items they contain

13. Make the bookshelves show how many documents they contain by inserting a line in the **VList** format statement in the **Format Features** section of the **Format** panel. The added line is shown highlighted below. The complete format statement can be copied from *sample_files* → *beatles* → *format_tweaks* → *show_num_docs.txt*.

```
<td valign=top>
{If}{[dc.Format] eq 'Audio',
[srclink][srcicon][/srclink],
{If}{[dc.Format] eq 'Images',
[srclink][thumbicon][/srclink],
[link][icon][/link]}}</td>
<td>{If}{[numleafdocs],([numleafdocs])}</td>
<td valign=top>[highlight]
{Or}{[dls.Title],[dc.Title],[Title],Untitled}
[/highlight]</td>
```

Preview the result (you don't need to build the collection.) Bookshelves in the titles and browse classifiers should show how many documents they contain.

Adding a Phind phrase browser

14. In the **Browsing Classifiers** section on the **Design** panel, add a **Phind** classifier. Leave the settings at their defaults: this generates a phrase browsing classifier that sources its phrases from *Title* and *text*.

Build the collection again and **preview** it. Select the new **Phrases** option from the navigation bar. Enter a single word in the text box, such as **band**. The phrase browser will present you with phrases found in the collection containing the search term. This can provide a useful way of browsing a very large collection. Note that even though it is called a phrase browser, only single terms can be used as the starting point for browsing.

Branding the collection with an image

15. To complete the collection, lets give it a new image for the top left corner of the page. Go to the **General** section of the **Format** panel. Use the browse button of **URL to 'about page' image:** to select the following image:

sample_files → beatles → advbeat_large → images → beatlesmm.png

Preview the collection, and make sure the new image appears.

Using UnknownPlug

In this section we incorporate the MIDI files. Greenstone has no MIDI plugin (yet). But that doesn't mean you can't use MIDI files!

16. **UnknownPlug** is a useful generic plugin. It knows nothing about any given format but can be tailored to process particular document types—like MIDI—based on their filename extension, and set basic metadata.

    In the **Document Plugins** section of the **Design** panel:

    - add **UnknownPlug**;
    - activate its **process_extension** field and set it to "mid" to make it recognize files with extension *.mid*;
    - Set **file_format** to "MIDI" and **mime_type** to "audio/midi".

    In this collection, all MIDI files are contained in the file *beatles_midi.zip*. **ZIPPlug** (already in the list of default plugins) is used to unpack the files and pass them down the list of plugins until they reach **UnknownPlug**.

17. **Build** the collection and **preview** it. Unfortunately the MIDI files don't appear as Audio under the *browse* button. That's because they haven't been assigned **dc.Format** metadata.

    - Back in the **Enrich** panel, click on the file *beatles_midi.zip* and assign its **dc.Format** value to "Audio"—do this by clicking on "Audio" in the **Existing values for dc. Format** list. All files extracted from the Zip file inherit its settings.

Cleaning up a title browser using regular expressions

*We now clean up the **Titles** browser.*

*To do this we must put the Librarian Interface into a different mode. The interface supports four levels of user: **Library Assistant**, who can add documents and metadata to collections, and create new ones whose structure mirrors that of existing collections; **Librarian**, who can, in addition, design new collections, but cannot use specialist IT features (e.g. regular expressions); **Library Systems Specialist**, who can use all design features, but cannot perform troubleshooting tasks (e.g. interpreting debugging output from Perl programs); and **Expert**, who can perform all functions.*

*So far you have mostly been operating in **Librarian** mode. We switch to **Library Systems Specialist** mode for the next exercise.*

18. To switch modes, click **File** → **Preferences...** → **Mode** and change to **Library Systems Specialist**. Note from the description that appears that you need to be able to formulate regular expressions to use this mode fully. That is what we do below.

19. Next we return to our **Titles** browser and clean it up. The aim is to amalgamate variants of titles

by stripping away extraneous text. For example, we would like to treat "ANTHOLOGY 1", "ANTHOLOGY 2" and "ANTHOLOGY 3" the same for grouping purposes. To achieve this:

- Go to the Title **AZCompactList** under **Browsing Classifiers** on the **Design** panel;
- Activate **removesuffix** and set it to:

```
(?i)(\\s+\\d+)|(\\s+[[:punct:]].*)
```

Build the collection and **preview** the result. Observe how many more times similar titles have been amalgamated under the same bookshelf. Test your understanding of regular expressions by trying to rationalize the amalgamations. (Note: *[[:punct:]]* stands for any punctuation character.) The icons beside the Word and PDF documents are not the correct ones, but that will be fixed in the next format statement.

*The previous exercise was done in **Library Systems Specialist** mode because it requires the use of regular expressions, something librarians are not normally trained in.*

*One powerful use of regular expressions in the exercise was to clean up the **Titles** browser. Perhaps the best way of doing this would be to have proper title metadata. The metadata extracted from HTML files is messy and inconsistent, and this was reflected in the original Titles browser. Defining proper title metadata would be simple but rather laborious. Instead, we have opted to use regular expressions in the **AZCompactList** classifier to clean up the title metadata. This is difficult to understand, and a bit fiddly to do, but if you can cope with its idiosyncrasies it provides a quick way to clean up the extracted metadata and avoid having to enter a large amount of metadata.*

Using non-standard macro files

To put finishing touches to our collection, we add some decorative features

20. Close the collection in the Librarian Interface (**File** → **Close**).

21. Using your Windows file browser outside Greenstone, locate the folder

    sample_files → beatles → advbeat_large

22. Open up another file browser, and locate the small beatles collection in your Greenstone installation:

    Greenstone → collect → smallbea

    smallbea is the folder name generated by Greenstone for this collection. You can determine what the folder name is for a collection by looking at the title bar of the Librarian Interface: the folder name is displayed in brackets after the collection name.

23. Using the file browser, copy the *images* and *macros* folders from the *advbeat_large* folder into the *smallbea* folder. (It's OK to overwrite the existing *images* folder: the image in it is included in the folder being copied.) The *images* folder includes some useful icons, and the *macros* folder defines some macro names that use these images.

    To see the macro definitions, open the collection in the Librarian Interface (**File** → **Open...**) and view the **Collection Specific Macros** section in the **Format** panel.

Using different icons for different media types

24. Re-edit your **VList** format statement to be the following (in **Format Features** on the **Format** panel). You can copy this text from the file *sample_files* → *beatles* → *format_tweaks* → *multi_icons.txt*.

```
<td valign=top>
  {If}{[numleafdocs],[link][icon][/link]}
  {If}{[dc.Format] eq 'Lyrics',[link]_iconlyrics_[/link]}
  {If}{[dc.Format] eq 'Discography',[link]_icondisc_[/link]}
  {If}{[dc.Format] eq 'Tablature',[link]_icontab_[/link]}
  {If}{[dc.Format] eq 'MARC',[link]_iconmarc_[/link]}
  {If}{[dc.Format] eq 'Images',[srclink][thumbicon][/srclink]}
  {If}{[dc.Format] eq 'Supplementary',[srclink][srcicon][/
srclink]}
  {If}{[dc.Format] eq 'Audio',[srclink]{If}{[FileFormat] eq 'MIDI',
_iconmidi_,_iconmp3_}[/srclink]}
</td>
<td>
{If}{[numleafdocs],([numleafdocs])}
</td>
<td valign=top>
[highlight]
{Or}{[dc.Title],[Title],Untitled}
[/highlight]
</td>
```

25. **Preview** your collection as before. Now different icons are used for discography, lyrics, tablature, and MARC metadata. Even MP3 and MIDI audio file types are distinguished. If you let the mouse hover over one of these images a "tool tip" appears explaining what file type the icon represents in the current interface language (note: *extra.dm* only defines English and French).

Changing the collection's background image

26. Go to the **Collection Specific Macros** section in the **Format** panel.

27. The content is fairly brief, specifying only what needs to be overridden from the default behaviour for this collection. Near the top you should see:

```
_collectionspecificstyle_ {
<style>
body.bgimage \{ background-image: url("_httpcimages_/beat_margin.
gif"); \}
\#page \{ margin-left: 120px; \}
</style>
}
```

Replace the text **beat_margin.gif** with **tile.jpg**.

This line relates to the background image used. The new image *tile.jpg* was in the *images* folder that was copied across previously.

28. **Preview** the collection's home page. The page background is now the new graphic.

    Other features can be altered by editing the macros—for example, the headers and footers used on each page, and the highlighting style used for search terms (specify a different colour, use bold etc.).

Building a full-size version of the collection

29. To finish, let's now build a larger version of the collection. To do this:

    - Close the current collection (**File** → **Close**).
    - Start a new collection called *large beatles* (**File** → **New...**).
    - Base this new collection on *small beatles*.
    - Copy the content of *sample_files* → *beatles* → *advbeat_large* → *import* into this newly formed collection. Since there are considerably more files in this set of documents the copy will take longer.
    - **Build** the collection and **preview** the result. (If you want the collection to have an icon, you will have to add it from the **Format** panel.)

Adding an image collage browser

30. Switch to the **Design** panel and select the **Browsing Classifiers** section. Pull down the **Select classifier to add** menu and select **Collage**. Click **<Add Classifier...>**. There is no need to customize the options, so click **<OK>** at the bottom of the resulting popup.

31. Now change to the **Create** panel and **build** and **preview** the collection.

7.3. Scanned image collection

Here we build a small replica of Niupepa, the Maori Newspaper collection, using five newspapers taken from two 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 "go to page" navigation feature at the top of the page.

*The collection involves a mixture of plugins, classifiers, and format statements. The bulk of the work is done by **PagedImgPlug**, a plugin 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. Three newspapers in our collection (all from the series "Te Whetu o Te Tau") 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**. Issue **ex.Volume** and **ex.Number** metadata is also recorded, where appropriate. 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.

2. In the **Gather** panel, open the *sample_files* → *niupepa* → *sample_items* folder 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: 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 **Document Plugins** section of the **Design** panel and delete **ImagePlug** and **TEXTPlug**.

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. Click **<OK>**.

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** and view one of the *Te Waka o Te Iwi* newspapers. Note that only the *Te Whetu o Te Tau* newspapers have text; *Te Waka o Te Iwi* papers don't.

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

*Under **Titles** documents from the same series are repeated without any distinguishing features such as date, volume or number. It would be better to group them by series title and display other information within each group. This can be accomplished using an **AZCompactList** classifier rather than **AZList**, and tuning the classifier's format statement.*

6. In the **Design** panel, under the **Browsing Classifiers** section, delete the **AZList** classifiers for **ex. Source** and **ex.Title**.

7. Now add an **AZCompactList** classifier, setting its **metadata** option to **ex.Title**, and add a **DateList** classifier, setting its **metadata** option to **ex.Date**.

8. **Build** the collection, and **preview** the *Titles* list and the *Dates* list.

9. Now we change the format statement for *Titles* to display more information about the documents. In the **Format Features** section of the **Format** panel, select the **ex.Title** classifier in the **Choose Feature** list, and **VList** in the **Affected Component** list. Click **<Add Format>** to add this format statement to your collection. Delete the contents of the **HTML Format String** box, and add the following text. (This format statement can be copied and pasted from the file *sample_files → niupepa → formats → titles_tweak.txt*.)

```
<td valign="top">[link][icon][/link]</td>
<td valign="top">
{If}{[numleafdocs],[ex.Title] ([numleafdocs]),
Volume [ex.Volume] Number [ex.Number] Date [ex.Date]}
</td>
```

10. Refresh in the web browser to view the new *Titles* list.

    As a consequence of using the **AZCompactList** classifier, bookshelf icons appear when titles are browsed. 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. For document nodes, Title is not displayed. Instead, Volume, Number and Date information are displayed.

11. The *Dates* list groups documents by date. A numeric date is displayed at the end of each document title, for example 18580601. This is in the Greenstone internal date format, which is crucial for the **DateList** classifier to correctly parse date metadata and generate an ordered date list. However, you can make the date look nice by adding a **[Format:]** macro to date metadata.

    In the **Format Features** section of the **Format** panel, select the **DateList** classifier. Replace the last line

    ```
    <td>{Or}{[dc.Date],[exp.Date],[ex.Date]}</td>
    ```

    with

    ```
    <td>{Or}{[dc.Date],[exp.Date],[format:ex.Date]}</td>
    ```

    Refresh in the web browser to view the new *Dates* list.

Displaying scanned images and suppressing dummy text

*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**.*

12. In the **Format Features** section of the **Format** panel, select the **DocumentText** format statement. The default format string displays the document's plain text, which, if there is none, is set to "This document has no text." Change this to the following text. (This format statement can be copied and pasted from the file *sample_files → niupepa → formats → doc_tweak.txt*)

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

    *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]`. This is a large image but it may be scaled by your browser.*

    This modification will display screenview image, but does nothing about the dummy text "This document has no text.", which will still be displayed. To get rid of this, edit the **DocumentText** format statement again and replace

    ```
    <td valign=top>[Text]</td>
    ```

    with

    ```
    {If}{[NoText] ne '1',<td valign=top>[Text]</td>}
    ```

13. **Preview** the collection and view one of the **Te Waka o Te Iwi** documents. The line "This document has no text." should now be gone.

Searching at page level

14. 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.

15. Go to the **Search Indexes** section of the **Design** panel. Remove the **ex.Source** index. Check the **section** checkbox to build the indexes on section level as well as document level. Make section level the default by selecting its **Default** radio button.

16. **Build** and **preview** the collection.

17. Set the display text used for the level drop-down menu by going to the **Search** section on the **Format** panel. Set the document level text to "newspaper", and the section level text to "page".

    Refresh in your web browser. Compare searching at "newspaper" level with searching at "page" level. A useful search term for this collection is "aroha".

18. 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. Go to **Format Features** section of the **Format** panel in the Librarian Interface, choose **All Features** in **Choose Feature** list, and select the **VList** format statement from the list of assigned format statements. Remove the second line from the **HTML Format String**:

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

While we are here, let's remove the filename from the display. Remove the following from the last line:

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

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

19. Now you will notice that page level search results only show the Title of the page (the page number), and not the Title of the newspaper. We'll modify the format statement to show the newspaper title as well as the page number. Also, lets add in Volume and Number information too.

    In the **Format Features** section, select **Search** in **Choose Feature**, and **VList** in **Affected Component**. Click **<Add Format>** to add this format to the collection. The previous changes modified **VList**, so they will apply to all **VList**s that don't have specific format statements. These next changes are made to **SearchVList** so will only apply to search results.

    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 text (it can be copied and pasted from the file *sample_files → niupepa → formats → search_tweak.txt*):

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

    Preview the search results. Items display newspaper title, Volume, Number and Date, and pages also display the page number.

*The collection you have just built involves a fairly complex document structure. There are two series of newspapers, **Te Waka** and **Te Whetu**.*

*In the **Te Waka** series there are two actual newspapers, Volume 1 Numbers 1 and 2. Number 1 has 4 pages, numbered 1, 2, 3, 4; Number 2 has 4 pages, numbered 5, 6, 7, 8. The page numbers increase consecutively through each volume, despite the fact that the volume is divided into different Numbers. Each page in the Te Waka series is represented by a single file, a GIF image of the page.*

The **Te Whetu** *series has three actual newspapers, Volume 1 Numbers 1, 2, and 3. Number 1 has 4 pages, numbered 1, 2, 3, 4; Number 2 has 5 pages, numbered 5, 6, 7, 8, 9; Number 3 has 5 pages, numbered 10, 11, 12, 13, 14. Again the page numbers increase consecutively through each volume. Each page in this series is represented by two files, a GIF image of the page and a text file containing the OCR'd text that appears on it.*

The key to this structure is in the respective .item *files. Here is a synopsis of the information they contain:*

```
(9-1-1) Te Waka Volume 1 Number 1
    p.1 gif
    p.2 gif
    p.3 gif
    p.4 gif
(9-1-2) Te Waka Volume 1 Number 2
    p.5 gif
    p.6 gif
    p.7 gif
    p.8 gif
(10-1-1) Te Whetu Volume 1 Number 1
    p.1 gif text
    p.2 gif text
    p.3 gif text
    p.4 gif text
(10-1-2) Te Whetu Volume 1 Number 2
    p.5 gif text
    …
    p.9 gif text
(10-1-3) Te Whetu Volume 1 Number 3
    p.10 gif text
    …
    p.14 gif text
```

7.4. Advanced scanned image collection

*In this exercise we build upon the collection created in the **Scanned image collection** exercise. We add a new newspaper by creating an item file for it, add a new newspaper using the extended XML item file format, and modify the formatting.*

Adding another newspaper to the collection

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

1. In the Librarian Interface, open up the Paged Image collection that was created in exercise **Scanned image collection** if it is not already open (**File** → **Open...**).

2. In the **Gather** panel, add the folder *sample_files* → *niupepa* → *new_papers* → *12* to your collection.

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

3. Create an item file for the collection. Have a look at an existing item file to see the format. Start up a text editor (e.g. WordPad) to open a new document. Add some metadata. The **Title** for this newspaper is "Te Haeata 1859-1862". The **Volume** is 3, **Number** is 6, and the **Date** is "18610902". (Greenstone's date format is **yyyymmdd**.) Metadata must be added in the form:

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

   For this document, the metadata looks like:

   ```
   <Title>Te Haeata 1859-1862
   <Date>18610902
   <Volume>3
   <Number>6
   ```

4. 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. You need to add a line for each page in the document. Make sure you increment the page number for each line.

5. Save the file using **Filename** *12_3_6.item*, and save as a plain text document. (If you are using Windows, make sure the file isn't saved as *12_3_6.item.txt*.) Back in the **Gather** panel of the Librarian Interface, locate the new file in the **Workspace** tree, and drag it into the collection, adding it to the **12** folder.

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

XML based item file

There are two styles of item files. The first, which was used in the previous section, uses a simple text based format, and consists of a list of metadata for the document, and a list of pages. This format allows specification of document level metadata, and a single list of pages.

The second style is an extended format, and uses XML. It allows a hierarchy of pages, and metadata specification at the page level as well as at the document level. In this section, we add in two newspapers which use XML-based item files.

7. In the **Gather** panel, add the folder *sample_files* → *niupepa* → *new_papers* → *xml* (you need to add the **xml** folder, not the **23** folder) to your collection.

8. Open up the file *xml* → *23* → *23__2.item* and have a look at the XML. This is **Number** *2* of the newspaper titled *Matariki 1881*. The contents of this document have been grouped into two sections: **Supplementary Material**, which contains an **Abstract**, and **Newspaper Pages**, which contains the page images (and OCR text).

9. **Build** and **preview** the collection. The xml style items have been included, but the document display for these items is not very nice.

Using process_exp to control document processing

10. Paged documents can be presented with a hierarchical table of contents, or with next and previous page arrows, and a "go to page" box (like we have done so far). The display type is specified by the **documenttype (hierarchy|paged)** option to **PagedImgPlug**. The next and previous arrows suit the linear sequence documents, while the table of contents suits the hierarchically organised document.

    Ordinarily, a Greenstone collection would have one plugin per document type, and all documents of that type get the same processing. In this case, we want to treat the XML-based item files differently from the text-based item files. We can achieve this by adding two **PagedImgPlug** plugins to the collection, and configuring them differently.

11. Change the mode in the Librarian Interface to **Library Systems Specialist** (or **Expert**) mode (using **File** → **Preferences...** → **Mode**), because you will need to change the order of plugins, and use regular expressions in the plugin options.

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

12. Go to the **Document Plugins** section of the **Design** panel, and add a new **PagedImgPlug** plugin. Enable the **screenview** option, set the **documenttype** option to **hierarchy** and set the **process_exp** option to **xml.*.item$**.

13. Move this **PagedImgPlug** plugin above the original one in the **Assigned Plugins** list.

14. The XML based newspapers have been grouped into a folder called *xml*. This enables us to process these files differently, by utilizing the **process_exp** option which all plugins support. The first **PagedImgPlug** in the list looks for item files underneath the *xml* folder. These documents will be processed as 'hierarchical' documents. Item files that don't match the process expression (i.e. aren't underneath the *xml* folder) will be passed onto the second **PagedImgPlug**, and these are treated as 'paged' documents.

    Rebuild and **preview** the collection. Compare the document display for a paged document e.g. **Te Waka o Te Iwi, Vol. 1, No. 1** with a hierarchical document, e.g. **Matariki 1881, No. 1**.

Switching between images and text

We can modify the document display to switch between the text version and the screenview and full size versions. We do this using a combination of format statements and macro files.

15. First of all we will add a macro file to the collection. Close the collection in the Librarian Interface. In a file browser outside of Greenstone, locate the Paged Image collection in your Greenstone installation: *Greenstone → collect → pagedima*.

    Also in a file browser, locate the file *sample_files → niupepa → macros → extra.dm*. Copy this file and paste it into the *macros* folder inside the pagedima collection.

16. Back in the Librarian Interface, open up the collection again, and go to the **Format Features** section of the **Format** panel.

17. Select **AllowExtendedOptions** in the **Choose Feature** list, and click **<Add Format>**. Tick the **Enabled** checkbox. This gives us more control over the layout of the page—in this case, we want to replace the standard *DETACH* and *NO HIGHLIGHTING* buttons with buttons that switch between images and text.

18. Select the **DocumentHeading** format item and set it to the following text (which can copied from *sample_files → niupepa → formats → adv_doc_heading.txt*).

    ```
    <div class="heading_title">{Or}{[parent(Top):ex.Title],[ex.Title]}
    </div>
    <div class="buttons" id="toc_buttons">
    {If}{[srcicon],_document:viewfullsize_}
    {If}{[screenicon],_document:viewpreview_}
    {If}{[NoText] ne '1',_document:viewtext_}
    </div>
    <div class="toc">[DocTOC]</div>
    ```

    `{Or}{[parent(Top):ex.Title],[ex.Title]}` outputs the newspaper Title metadata. This is only stored at the top level of the document, so if we are at a subsection, we need to get it from the top (`[parent(Top):ex.Title]`). Note that we can't just use `[parent:ex.Title]` as this retrieves the Title from the immediate parent node, which may not be the top node of the document.

    `_document:viewpreview_, _document:viewfullsize_, _document:viewtext_` are macros defined in *extra.dm* which output buttons for preview, fullsize and text versions, respectively. We choose which buttons to display based on what metadata and text the document has. Note you can view the macros by going to the **Collection Specific Macros** section of the

Format panel.

`[DocTOC]` is the document table of contents or "go to page" navigation element. Since we are using extended options, we need to explicitly specify this for it to appear in the page.

The different pieces are surrounded by `<div>` elements, so that the appropriate styling information can be used.

19. Select the **DocumentText** format statement and set it to the following text (which can be copied from *sample_files* → *niupepa* → *formats* → *adv_doc_text.txt*):

    ```
    {If}{_cgiargp_ eq 'fullsize',[srcicon],
    {If}{_cgiargp_ eq 'preview',[screenicon],
    {If}{[NoText] ne '1',[Text],[screenicon]}}}
    ```

    This format statement changes the display based on the "p" argument (`_cgiargp_`). This is not used normally for document display, so we can use it here to switch between full size image (`[srcicon]`), preview size image (`[screenicon]`) and text (`[Text]`) versions of each page.

20. **Preview** the collection. View some of the documents—once you have reached a newspaper page, you should get fullsize, preview and text options.