Lab 7: Scanned images

7.1. 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 **PagedImagePlugin**, 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 **PagedImagePlugin** 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 **PagedImagePlugin**, not **ImagePlugin** or **TextPlugin**. Switch to the **Document Plugins** section of the **Design** panel and *delete **ImagePlugin** and **TextPlugin***.

4. Open up the configuration window for **PagedImagePlugin** by double-clicking on the plugin. Switch on its **create_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 **create_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 (CL1) 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 (CL2) 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.

12. Now we format the date. In the **Format Features** section of the **Format** panel, select the **DateList** classifier *and set **Affected Component** to **DateList***. 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**.*

13. 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 **PagedImagePlugin**. 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>}
```

14. **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

15. 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.

16. 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.

17. **Build** and **preview** the collection.

18. 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".

19. 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>
    ```

    The reason why this is causing a problem is that the **PagedImagePlugin** does not produce ex.thumbicon, and as a consequence this format statement displays ex.srcicon, which is very large

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

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

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

20. 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
```

7.2. 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 **PagedImagePlugin**. 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 **PagedImagePlugin** 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 **PagedImagePlugin** plugin. Enable the **create_screenview** option, set the **documenttype** option to **hierarchy** and set the **process_exp** option to **xml.*.item$**.

13. Move this **PagedImagePlugin** 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 **PagedImagePlugin** 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 **PagedImagePlugin**, 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.
