LAB 3:

GSDL: Advanced collection configuration

3.1. 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 --**.

2. 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.*

3. 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 "5 documents were processed and included in the collection. 1 was rejected".

4. 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.*

5. 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 pdf05notext.pdf to HTML format". pdftohtml.pl cannot convert a PDF file to HTML if the PDF file has no extractable text.

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

Splitting PDFs into sections

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

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

9. **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

1. 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.

2. In the **Document Plugins** section, configure **PDFPlugin**. 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.

3. **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.

4. 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]
   ```

5. 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,[screenicon],[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)

6. 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.
7. We achieve this by putting the problem files into a separate folder, and adding another **PDFPlugin** plugin with different options.
8. 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>**.
9. 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.

10. For versions before 2.82 you need to change to **Library Systems Specialist** mode in order to 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.

11. Switch to the **Document Plugins** section of the **Design** panel. Add a second PDF plugin by selecting **PDFPlugin** 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>**.
12. Configure the first PDF plugin, and set the **process_exp** option to **'notext.*\.pdf'**.
13. The two PDF plugins should have options like the following:

```
plugin PDFPlugin -convert_to pagedimg_jpg -process_exp 'notext.*\.pdf'
plugin PDFPlugin -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 **PDFPlugin** will process any PDF files in the *notext* directory. The second **PDFPlugin** 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. For versions before 2.82 this option is only visible in **Library Systems Specialist** and **Expert** modes.

For versions before 2.82, change back to **Librarian** mode.

14. 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 `{Or}{[srcicon],[Text]}`.
15. 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 "banana" or "METS").

Opening PDF files with query terms highlighted

16. 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.
17. 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.

    PDFPlugin 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 **PDFPlugin** 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.)

18. Add **SearchVList** 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. 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;\">{Or}{[ex.thumbicon],[ex.srcicon]}</a>,
    [ex.srclink]{Or}{[ex.thumbicon],[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. Enhanced Word document handling

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

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

Using Windows native scripting

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

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

3. **Build** the collection. You will notice that the Microsoft Word program is started up for each Word document—the document is saved as HTML from Word itself, to get a better conversion. **Preview** the collection. In the **Titles** list, notice that *word03.doc* and *word06.doc* now have a book icon, rather than a page icon. These now appear with hierarchical structure.

   The default behaviour for **WordPlugin** with **windows_scripting** is to section the document based on "Heading 1", "Heading 2", "Heading 3" styles. If you open up the *word03.doc* or *word06.doc* documents in Word, you will see that the sections use these Heading styles.

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

4. Some of the documents do not use styles (e.g. *word01.doc*) and no structure can be extracted from them. Some documents use user-defined styles. **WordPlugin** can be

configured to use these styles instead of Heading 1, Heading 2 etc. Next we will configure **WordPlugin** to use the styles found in *word05.doc*.

Modes in the Librarian Interface

5. Since versions 2.82, the Librarian Interface can operates in three modes, while for versions before 2.82 GLI operates in four modes. Go to **File → Preferences... → Mode** and see the modes and what functionality they provide access to. **Librarian** is the default mode.

6. For versions before 2.82, change the mode to **Library Systems Specialist** because you will need to use regular expressions to set up the style options in the next part of the exercise.

Defining styles

7. Open up *word05.doc* in Word (by double-clicking on it in the **Gather** pane), and examine the title and section heading styles. You will see that various user-defined header styles are set such as:
   - *ManualTitle*: Title of the manual
   - *ChapterTitle:* Level 1 section heading
   - *SectionHeading*: Level 2 section heading
   - *SubscriptionHeading*: Level 3 section heading
   - *AppendixTitle*: Appendix section title

8. In the **Document Plugins** section of the **Design** panel, select **WordPlugin** and click **<Configure Plugin...>**. Four types of header can be set which are:
   - `level1_header (level1Header1|level1Header2|...)`
   - `level2_header (level2Header1|level2Header2|...)`
   - `level3_header (level3Header1|level3Header2|...)`
   - `title_header (titleHeader1|titleHeader2|...)`

   These header options define which styles should be considered as title, level 1, level 2 and level 3 styles.

   Ensure that the **windows_scripting** option is checked, and set the options as follows (spaces in the Word styles are removed when converting to HTML styles, and these options must match the HTML styles):

   ```
   level1_header:(ChapterTitle|AppendixTitle)
   level2_header: SectionHeading
   level3_header: SubjectHeading
   title_header: ManualTitle
   ```

   Once these are set, click **<OK>**.

9. Close any documents that are still open in Word, as this can prevent the build process from completing correctly.

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

    If you have documents with different formatting styles, you can use `(...|...)` to specify all of the different styles.

Removing pre-defined table of contents

11. If you look at *word06.doc* you will see that it now has two tables of contents. One is generated by Greenstone based on the document's styles, the other was already defined in the Word document. **WordPlugin** can be configured to remove predefined tables of contents and tables of figures. The tables must be defined with Word styles in order for this to work.

12. To remove the tables of contents and figures from *word06.doc*, switch on the **delete_toc** option in **WordPlugin**. Set the **toc_header** option to **`(MsoToc1|MsoToc2|MsoToc3|MsoTof|TOA)`.** In this document, the table of contents and list of figures use these four style names. Click **<OK>**.

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

14. For versions before 2.82, , switch the Librarian Interface back to **Librarian** mode (**File → Preferences... → Mode**).

Extracting document properties as metadata

15. When the **windows_scripting** option is set, word document properties can be extracted as metadata. By default, only the Title will be extracted. Other properties can be extracted using the **metadata_fields** option.

16. In the **Enrich** panel, look at the metadata that has been extracted for *word05.doc* and *word06.doc*. Now open the documents in Word and look at what properties have been set (**File → Properties**). They have Title, Author, Subject, and Keywords properties. **WordPlugin** can be configured to look for these properties and extract them.

17. In the **Design** panel, under **Document Plugins**, configure **WordPlugin** once again. Switch on the configuration option **metadata_fields**. Set the value to

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

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

18. Make sure you have closed all the documents that were opened, then **rebuild** the collection.

19. Look at the metadata for the two documents again in the **Enrich** panel. You should now see ex.Creator and ex.Subject metadata items. This metadata can now be used in display or browsing classifiers etc.

3.4. A large collection of HTML files—Tudor

1. Invoke the Greenstone Librarian Interface (from the Windows *Start* menu) and start a new collection called **tudor** (use the **File** menu), based on the default **-- New Collection --**.

2. In the **Gather** panel, open the *tudor* folder in *sample_files*.

3. Drag *englishhistory.net* from the left-hand side to the right to include it in your **tudor** collection. (This material is from Marilee Hanson's Tudor England Collection at http://englishhistory.net/tudor.html, distributed with her permission.)

4. Switch to the **Create** panel and click **<Build Collection>**.

5. When building has finished, **preview** the collection.

Extracting more metadata from the HTML

6. The browsing facilities in this collection (*Titles* and *Filenames*) are based entirely on extracted metadata. Return to the **Enrich** panel in the Librarian Interface and examine the metadata that has been extracted for some of the files.

7. Many HTML documents contain metadata in `<meta>` tags in the `<head>` of the page. Open up the *englishhistory.net* → *tudor* → *monarchs* → *boleyn.html* file by navigating to it in the tree on the left hand side, and double clicking it. This will open it in a web browser. View the HTML source of the page (**View** → **Source** in Internet Explorer, **View** → **Page Source** in Mozilla). You will notice that this page has *page_topic, content* and *author* metadata.

8. By default, **HTMLPlugin** only looks for Title metadata. Configure the plugin so that it looks for the other metadata too. Switch to the **Design** panel and select the **Document Plugins** section. Select the **plugin HTMLPlugin** line and click **<Configure Plugin...>**. A popup window appears. Switch on the **metadata_fields** option, and set the value to

```
Title,Author,Page_topic,Content
```

Make sure that you have copied this exactly, with no spaces. Click **<OK>**.

9. Switch to the **Create** panel and **rebuild** the collection. Go back to the **Enrich** panel and look at the extracted metadata for some of the HTML files in *englishhistory.net → tudor → monarchs*. The new metadata should now be visible.

Looking at different views of the files in the Gather and Enrich panels

12. Switch to the **Gather** panel and in the right-hand side open *englishhistory.net → tudor*.

13. Change the **Show Files** menu for the right-hand side from **All Files** to **HTM & HTML**. Notice the files displayed above are filtered accordingly, to show only files of this type.

14. Change the **Show Files** menu to **Images**. Again, the files shown above alter.

15. Now return the **Show Files** setting back to **All Files**, otherwise you may get confused later. Remember, if the **Gather** or **Enrich** panels do not seem to be showing all your files, this could be the problem.

3.5. Enhanced collection of HTML files—Tudor

We return to the Tudor collection and add metadata that expresses a subject hierarchy. Then we build a classifier that exploits it by allowing readers to browse the documents about Monarchs, Relatives, Citizens, and Others separately.

Adding hierarchically-structured metadata and a Hierarchy classifier

1. Open up your **tudor** collection, switch to the **Enrich** panel and select the *citizens* folder (a subfolder of *englishhistory.net → tudor*). Set its **dc.Subject and Keywords** metadata to **Tudor period|Citizens**. The vertical bar ("|") is a hierarchy marker. Selecting a *folder* and adding metadata has the effect of setting this metadata value for all files contained in this folder, its subfolders, and so on. A popup alerts you to this fact. Click **<OK>** to close the popup.

2. Repeat for the *monarchs* and *relative* folders, setting their **dc.Subject and Keywords** metadata to **Tudor period|Monarchs** and **Tudor period|Relatives** respectively. Note that the hierarchy appears in the **Existing values for dc.Subject and Keywords** area.

   If you don't want to see the popup each time you add folder level metadata, tick the **Do not show this warning again** checkbox; it won't be displayed again.

3. Finally, select all remaining files—the ones that are not in the *citizens*, *monarchs*, or *relative* folders—by selecting the first and shift-clicking the last. Set their **dc.Subject and Keywords** metadata to **Tudor period|Others**: this is done in a single operation (there is a short delay before it completes).

When multiple files are selected in the left hand collection tree, all metadata values for all files are shown on the right hand side. Items that are common to all files are displayed in black—e.g. **dc.Subject and Keywords**—while others that pertain to only one or some of the files are displayed in grey—e.g. any extracted metadata.

Metadata inherited from a parent folder is indicated by a folder icon to the left of the metadata name. Select one of the files in the *relative* folder to see this.

4. Switch to the **Design** panel and select **Browsing Classifiers** from the left-hand list. Set the menu item for **Select classifier to add:** to **Hierarchy**; then click **<Add Classifier...>**.

5. A window pops up to control the classifier's options. Change the **metadata** to **dc.Subject and Keywords** and then click **<OK>**.

6. For tidiness' sake, **remove** the **classifier** for **Source** metadata (included by default) from the list of currently assigned classifiers, because this adds little to the collection.

7. Now switch to the **Create** panel, **build** the collection, and **preview** it. Choose the new **Subjects** link that appears in the navigation bar, and click the bookshelves to navigate around the four-entry hierarchy that you have created.

Adding a hierarchical phrase browser (PHIND)

Next we'll add an interactive hierarchical phrase browsing classifier to this collection.

8. Switch to the **Design** panel and choose the **Browsing Classifiers** item from the left-hand list.

9. Choose **Phind** from the **Select classifier to add:** menu. Click **<Add Classifier...>**. A window pops asking for configuration options: leave the values at their preset defaults (this will base the phrase index on the full text) and click **<OK>**.

10. **Build** the collection again, **preview** it, and try out the new **Phrases** option in the navigation bar. An interesting PHIND search term for this collection is "king". Note that even though it is called a phrase browser, only single terms can be used as the starting point for browsing.

Partitioning the full-text index based on metadata values

*Next we partition the full-text index into four separate pieces. To do this we first define four subcollections obtained by "filtering" the documents according to a criterion based on their **dc.Subject and Keywords** metadata. Then an index is assigned to each subcollection. This will enable users to restrict a search to a subset of the documents.*

11. Switch to the **Design** panel, and click **Partition Indexes**. For versions before 2.82, this feature is disabled because you are operating in **Librarian** mode (this is indicated in the title bar at the top of the window).

12. Switch to **Library Systems Specialist** mode by going to **Preferences...** (on the **File** menu) and clicking **<Mode>**. Read about the other modes too.

13. Return to the **Partition Indexes** section of the **Design** panel. Ensure that the **Define Filters** tab is selected (the default). Define a subcollection filter with name **monarchs** that matches against **dc.Subject and Keywords**, and type **Monarchs** as the regular expression to match with. Click **<Add Filter>**. This filter includes any file whose **dc.Subject and Keywords** metadata contains the word *Monarchs*.

14. Define another filter, **relatives**, which matches **dc.Subject and Keywords** against the word **Relatives**. Define a third and fourth, **citizens** and **others**, which matches it against the words **Citizens** and **Others** respectively.

15. Having defined the subcollection filters, we partition the index into corresponding parts. Click the **Assign Partitions** tab. Select the citizens subcollection and click **<Add Partition>**. Next select monarchs, and click **<Add Partition>**. Repeat for the other two subcollections, so that you end up with four partitions, one based on each subcollection filter.

    The order they appear in the **Assigned Subcollection Partitions** list is the order they will appear in the drop down menu on the search page. You can change the order by using the **<Move Up>** and **<Move Down>** buttons.

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

17. The search page includes a pulldown menu that allows you to select one of these partitions for searching. For example, try searching the *relatives* partition for *mary* and then search the *monarchs* partition for the same thing.

5. To allow users to search the collection as a whole as well as each subcollection individually, return to the **Partition Indexes** section of the **Design** panel and select the **Assign Partitions** tab. Select all four subcollections by checking their boxes and click **<Add Partition>**.

19. To ensure that the combined index appears first in the list on the reader's web page, use the **<Move Up>** button to get it to the top of the list here in the **Design** panel. Then **build** and **preview** the collection.

20. Search for a common term (like *the*) in all five index partitions, and check that the numbers of words (not documents) add up.

21. The text in the drop down box on the search page is based on the filters each partition was built on. To change the text that is displayed, go to the **Search** section of the **Format** panel. The single filter partitions have sensible default text, but the combined partition does not. Set the **Display text** for the combined partition to "all". **Preview** the collection.

22. For versions before 2.82, now return to **Librarian** mode in the Librarian Interface, using **Preferences...** (on the **File** menu).

Controlling the building process

*Finally we look at how the building process can be controlled. Developing a new collection usually involves numerous cycles of building, previewing, adjusting some enrich and design features, and so on. While prototyping, it is best to temporarily reduce the number of documents in the collection. This can be accomplished through the **maxdocs** parameter to the building process.*

23. Switch to the **Create** panel and view the options that are displayed in the top portion of the screen. Select **maxdocs** and set its numeric counter to **3**. Now **build**.

24. Preview the newly rebuilt collection's **Titles** page. Previously this listed more than a dozen pages per letter of the alphabet, but now there are just three—the first three files encountered by the building process.
25. Go back to the **Create** panel and turn off the **maxdocs** option. **Rebuild** the collection so that all the documents are included.

3.6. 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>
```

3. **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.*

4. 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>
```

5. **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.

6. 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>
```

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.)

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

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

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

3.7. 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 **HTMLPlugin** 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.

Let us create a new collection for ebooks, based on tagged html files given in the sample files.

1. Start a new collection called **Ebooks Collection**.

2. Drag and drop the *tobacoebook* folder from *sample_files → IIMK_Sample Files* into the collection area in the **Gather** Panel.

3. In the **Document Plugins** section of the **Design** panel, note that **HTMLPlugin** has the **description_tags** option set. This option is needed when `<Section>` tags are used in the source documents.

4. In the **Format Panel** select the **DocumentImages** option from the **Choose Feature**, **Add Format** and make it **Enabled.** (This option is for getting a cover image for the ebook. Please note, file name for the cover image and the html file should be same.)

5. **Build** and **preview** the collection.

5.3. 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.

****