percs Documentation

Release 1.0.0

Chris Nilsson

Aug 03, 2018

Contents

Contents:

CHAPTER 1

About

This is the code & project documentation for the percs website.

Percs was initially created to provide a basic, searchable index of the NSW Pecuniary interest documents. These files are uploaded as large PDFs, containing scanned images of Minister's submissions. Nearly 100 submissions, dumped into big, unsearchable, opaque files.

One OpenAustralia pub meetup, Luke Bacon kicked off the effort by processing & splitting the files into individual per-minister submissions.

Chris Nilsson ran these files through an OCR, indexed the found text, and wrapped this percs site around the result.

The aim is to digitise & index previous, and ongoing years as needed.

Why? These folks are in charge of a good chunk of our money. Having their pecuniary interest declarations more accessible can only help keep things transparent and fair.

Of course, the process isn't perfect. Many documents are handwritten, and difficult enough for humans to read, letalone the OCR.

But, it's a start.

1.1 Source code

You can get the source code from: https://github.com/otherchirps/percs

1.2 Documentation

Usage instructions, etc, can be found here: http://percs.readthedocs.org/

1.3 Licensing

The site itself, and its custom libraries are available at no charge, and licensed under the Mozilla Public License 2.0. The third party libraries (which actually do the cool stuff) have their own licenses.

PDFs sourced from the NSW State Government, and are Copyright © State of New South Wales (NSW Parliament). To the best of my knowledge, I'm complying with their terms of use.

1.4 Contact

Site-specific suggestions and problems should go to the issue tracker.

Any other queries, you can reach me via email.

CHAPTER 2

Usage

To search the content, first we need to:

1. Prepare the PDF files (OCR, split into per-person files), then
2. Add the text content to a search index.

These can be resource-consuming operations for little webservers, so in this *Implementation*, we perform these steps offline. That is, on your local workstation or wherever. Just not on the server.

It's feasible that the indexing could be performed by the server, as that's not too taxing, given percs doesn't do anything too fancy with the content (just using the Whoosh stemming analyzer). But as the volume of updates is minimal, it was quicker to get going with the server treating the index as static.

Also be aware that the tools included here are still fairly task-specific. In that they didn't start life as generalised tools. They've been coaxed along that path towards generalisation a bit, but they still leak their heritage in places.

2.1 Installation

2.1.1 Percslib

This is the main support library that coordinates all the pdf manipulation, indexing and search facilities.

It's found in the `/percslib` subdirectory.

The OCR library has a bunch of dependencies. On Ubuntu 14.10, I installed these packages to get things rolling:

```
sudo apt-get install libpython-dev zlibc zlib1g zlib1g-dev libtk8.6 libjpeg-dev␣
↪libopenjpeg-dev libfreetype6-dev libwebp-dev libwebpmux1 liblcms2-2 liblcms2-dev␣
↪liblcms2-utils libtiff5-dev tesseract-ocr ghostscript imagemagick libpoppler-dev␣
↪libpoppler-cil-dev python-poppler
```

Unpack the repository. Use a virtualenv if you like (probably a good idea). Then, in the `percslib` subdirectory, run:

```
python setup.py install
```

2.1.2 Website

This is a web2py application. So you'll need to install web2py separately before the code in `/website` will work. Once you've installed web2py somewhere, link / copy the percs website contents to:

```
[web2py install location]/applications/percs
```

Alternatively, it's should be fairly trivial to add a website in whichever framework you prefer, as long as it can call the percslib code (either by importing it as a python module, or calling the commandline scripts).

Why did I go for web2py? I *like* web2py.

Collections

The PDF files we provide are kept on the filesystem, under:

```
[web2py instance]/applications/percs/static/collections/[COLLECTION NAME]/
```

Where `COLLECTION NAME` matches the collection the files were indexed under (more on that in *Indexing*). The filenames must also match those they were added to the index with. Otherwise, the search result hit links won't work.

Note: These naming requirements are only an implementation detail of the current website. In keeping with the notes in *Implementation*, this is an attempt at avoiding the need for more infrastructure.

Each collection directory can contain a `README` file, with a description to display on the collection's page. The contents of this README should be in MARKMIN format.

2.2 Preprocessing

Percs included two main PDF preprocessing functions:

1. Adding a text layer via OCR. This relies on the fantastic PyPDFOCR library.
2. Splitting subsets of pages into separate PDF files. This relies on the PyPDF2 library, which is a dependency of PyPDFOCR anyway.

2.2.1 Adding an OCR text layer

Percs provides a commandline script to do this, but it delegates the work entirely to PyPDFOCR.

```
$ percs-pdf-ocr -h
usage: percs-pdf-ocr [-h] [-d] [-v] [-m] [-l LANG] [--skip-preprocess]
                     [-w WATCH_DIR] [-f] [-c CONFIGFILE] [-e] [-n]
                     [pdf_filename]

Convert scanned PDFs into their OCR equivalent. Depends on GhostScript and
Tesseract-OCR being installed.
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  pdf_filename          Scanned pdf file to OCR

optional arguments:
  -h, --help            show this help message and exit
  -d, --debug           Turn on debugging
  -v, --verbose         Turn on verbose mode
  -m, --mail            Send email after conversion
  -l LANG, --lang LANG  Language(default eng)
  --skip-preprocess     Skip preprocessing (saves time) if your pdf is in good
                        shape already
  -w WATCH_DIR, --watch WATCH_DIR
                        Watch given directory and run ocr automatically until
                        terminated

Filing optinos:
  -f, --file            Enable filing of converted PDFs
  -c CONFIGFILE, --config CONFIGFILE
                        Configuration file for defaults and PDF filing
  -e, --evernote        Enable filing to Evernote
  -n                    Use filename to match if contents did not match
                        anything, before filing to default folder

PyPDFOCR version 0.8.2 (Copyright 2013 Virantha Ekanayake)
```

As you can see, it's simple a thin-wrapper for PyPDFOCR's commandline script.

An example of using this:

```
$ percs-pdf-ocr 2014-06-30_O-Farrell_Barry_pecuniary-interests.pdf
```

This will create a new file, with an "_ocr" suffix added to the filename.

PyPDFOCR has a tonne of other features (monitoring a directory for new files, automatically filing the input & output files after processing – based on text found during OCR'ing, sending notification emails. . .). It's pretty handy.

2.2.2 Splitting PDF Files

One main problem with the OCR is that in our current target files, they contain around 100 member submissions. Many of these submissions are hand written, so hardly any useful content can be extracted.

The fallback is to split these mega-documents into the individual submissions. So even if we failed to find any searchable content, we can still search / locate the submission by the person's name, and a human can eyeball the text that the machine couldn't read. (The human might also have trouble. . . some of the hand writing is pretty poor.)

The good news

Percs supplies a script, which accepts a CSV of:

1. start page (with 1 being the first page in the file),
2. end page,
3. document name (eg. the person's name),
4. output filename for this page range.

Here's it's help dump:

```
$ percs-pdf-split -h
usage: percs-pdf-split [-h] [-c CSV_FILENAME] [-o OUTPUT_DIR] [-H] filename

positional arguments:
filename              PDF file to split

optional arguments:
-h, --help            show this help message and exit
-c CSV_FILENAME, --csv-page-ranges CSV_FILENAME
                       CSV containing start, end, name & (optional) output
                       filename for range
-o OUTPUT_DIR, --output-directory OUTPUT_DIR
                       Output directory. Default: current directory
-H, --header-row      CSV contains header row to skip
```

Example usage, here I split the 2012-2013 Volume 2 file into it's individual submissions, with the files being dumped to the `split_v2` directory:

```
percs-pdf-split -c 2012-2013_perc_interests_split_v2.csv -o splits_v2 "Volume 2 -␣
↪Disclosures by Members - 30 June 2013_ocr.pdf"
```

Here are the first couple of rows of the csv:

```
3,11,Issa_Tony,2013-06-30-Issa_Tony-Nsw_2012-2013_pecuniary_interests.pdf
12,20,Kean_Matthew,2013-06-30-Kean_Matthew-Nsw_2012-2013_pecuniary_interests.pdf
21,29,Lalich_Nick,2013-06-30-Lalich_Nick-Nsw_2012-2013_pecuniary_interests.pdf
```

The document name that you provide will also be embedded in the resulting PDF's "Subject" metadata property. Maybe this is useful if you index the files with other software as well.

The bad news

So far, I haven't found a nice automatic way of determining the page ranges. Creating the page range CSV is still a manual slog.

Ideas for solving this are welcome!

2.3 Indexing

Now we have the OCR'd individual PDF files ready, so we can create an index. Or, if there's an existing index, we can add these documents to it.

For this, there's the `percs-pdf-index` script:

```
$ percs-pdf-index -h
usage: percs-pdf-index [-h] [-i INDEX_DIR] [-f] [-c COLLECTION] [-n NAME_CSV]
                       [-r] [-s] [-d DELETE_DOC]
                       directory

positional arguments:
  directory             Directory path containing PDFs to be indexed

optional arguments:
```

(continues on next page)

(continued from previous page)

```
  -h, --help            show this help message and exit
  -i INDEX_DIR, --index-dir INDEX_DIR
                        Index directory. Default: CURRENT_DIR/percs_idx
  -f, --force-new       If index dir is an existing index, clear and create a
                        new index in its place
  -c COLLECTION, --collection-name COLLECTION
                        Collection name for indexed files. Default to current
                        date (yyyy-mm-dd)
  -n NAME_CSV, --name-csv NAME_CSV
                        Load names from csv, instead of embedded pdf metadata.
                        Expects (name, filename) rows
  -r, --remove-missing  Remove missing files from the index, when indexing a
                        directory
  -s, --size            Show number of records in index, and exit
  -d DELETE_DOC, --delete-document DELETE_DOC
                        Delete file from index, and exit
```

Example usage, adding the earlier `splits_v2` directory we used for `percs-pdf-split`:

```
$ percs-pdf-index -i percs_idx -c "nsw_2012-2013_pecuniary_interests" splits_v2
```

This will add the PDFs, in `splits_v2`, to the index in `percs_idx`, under the collection named `nsw_2012-2013_pecuniary_interests`.

2.3.1 Document names

By default, this will extract the document name from the embedded "Subject" field, of the PDFs created by percs-pdf-split. Otherwise, if the files come from another source & don't have this, you can supply another CSV (–name-csv). This CSV expects the format:

1. name (eg. person name),
2. filename

2.4 Searching

A commandline search utility is included: `percs-search`. You point it at your index directory, and give it a query, using the Whoosh default query language. Results will be spat out as json chunks.

Options:

```
$ percs-search -h
usage: percs-search [-h] [-i INDEX_DIR] [-p PAGE] [-l LIMIT] [-c] query

positional arguments:
  query                 Query string

optional arguments:
  -h, --help            show this help message and exit
  -i INDEX_DIR, --index-dir INDEX_DIR
                        Percs index to search against
  -p PAGE, --page PAGE  Page of paginated results to return
  -l LIMIT, --limit LIMIT
```

(continues on next page)

(continued from previous page)

```
                        Max number of results to return
  -c, --contents        Return indexed contents with each record
```

Example:

```
percs-search -i percs_idx 'golf club' --limit 1
```


```
{
  "matches": [
    {
      "file_hash": "b3d90a4d975791aef8719e89ee9f36b2f538d93d83bd5e2ee3283f6f71d1373c",
      "highlights": "<b class=\"match term0\">Club</b>,\n\nQantas Frequent Flyer <b
↪class=\"match term0\">Club</b>, Port Kembla Pumas, Angels...of Hope, Wollongong <b
↪class=\"match term1\">Golf</b>\n\nI <b class=\"match term0\">Club</b>, Illawarra
↪Stingrays...Social <b class=\"match term0\">Club</b>, Member of Dogs NSW\n\n22",
      "page_hash": "16fbe6442983119358f0c21e318564c749a2ea9959e3ce3f8e212d3f47c66b42",
      "collection": "nsw_2013-2014_pecuniary_interests",
      "filename": "2014-06-30_Hay_Noreen_pecuniary-interests_ocr.pdf",
      "person": "Hay_Noreen",
      "content_type": "application/pdf",
      "id": "b3d90a4d975791aef8719e89ee9f36b2f538d93d83bd5e2ee3283f6f71d1373c-9",
      "page": 9
    }
  ],
  "total_matches": 5,
  "matches_returned": 1,
  "query": "golf club"
}
```

CHAPTER 3

Implementation

The original PDFs were run through Tesseract, using PyPDFOCR. The text content was then indexed using PDFMiner, Whoosh.

OCR and indexing is done offline. The static index is then served by the website.

The percslib python package can, and probably should, be split off into its own project, as it can split, index, and search independantly of the little website tacked in front of it. Maybe one day...

3.1 Why do it this way?

When there are very cool open source search platforms, like Elasticsearch and Solr, which *already* index PDFs out of the box?

Hosting cost.

The aim was to not only index the text, but make it available for near zero cost. Servers which can handle the OCR processing (lots of CPU...), or an Elasticsearch/Solr instance (lots of memory...), don't cost near to nothing.

The current implementation means it can be thrown onto tiny shared hosts without a worry.

Plus, Whoosh kicks ass!

CHAPTER 4

Indices and tables

- genindex
- modindex
- search