Cloudy biographies and portrait walls

Author : Tim Sherratt

Tagged as : ADB Online, biographies, Cooliris, python, visualisation, word clouds

Date : January 24, 2009

With a bit of time to play over Christmas I had a go at applying some of the techniques described at *ProgrammingHistorian* to the ADB Online. I thought it might be interesting to create some word clouds, both for what they could reveal about the content of the ADB, and to see what they had to offer as a way of improving access to the articles.

So I set about learning Python and was soon downloading and scraping the more than 10,000 articles that make up the ADB online.

My first tests revealed that the most frequent words in ADB articles were…

born and **died**

Who'd have thought it? In a biographical dictionary?

After further refining the stopwords list I started to generate some useful clouds. Finally after 147 minutes of processing time, I had a word cloud representing the content of all 16 volumes of the *Australian Dictionary of Biography*.

The complete ADB word cloud

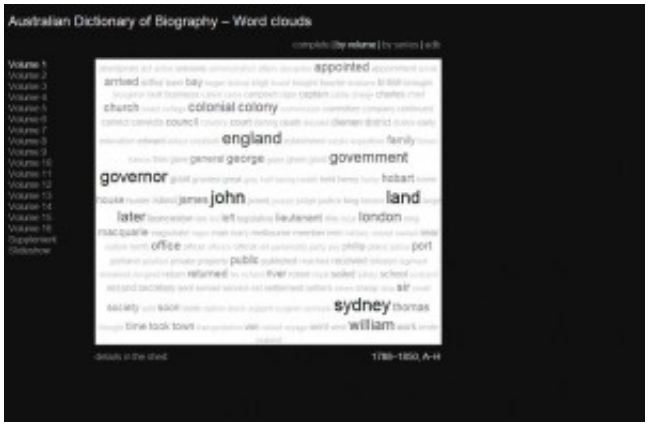
The words in the cloud are linked back to the ADB's own search engine, allowing the cloud to be used as a way of exploring the articles themselves.

It shows the top 200 words, but if you want to see the rest you can download the raw word frequency file (>1mb txt file).

What can you see? Amongst other things, John is obviously the most popular name, Sydney just edges out Melbourne as the most popular place, and burial beats cremation as the most common

mode of dispatch. It's fun to explore.

But of course this then set me wondering about how these frequencies might change with the development of the ADB and changes in its subjects. So I generated word clouds for each volume and for each chronological series.



Word clouds by volume

Word clouds by series

I even added some simple Javascript slideshows so you could watch the clouds evolve.

One of the most obvious features in the series clouds is the gradual disappearance of 'land'. It's one of the most prominent words in the first series, but gradually fades until it disappears completely in the last.

After this successful foray into the world of word clouds, I began to think about other ways of visualising the ADB's content. Many of the articles have portrait images, wouldn't it be interesting to use the images themselves as the entry point to the biographical articles?

I'd already been playing with CoolIris, so I decided to harvest all the portrait references and use them to create a 3D wall. The result is pretty spectacular.

ADB portrait browser

Some technical details about the clouds and the portrait browser follow, for those interested in such things...

Gathering your words

Conveniently for me, *ProgrammingHistorian* uses the *Dictionary of Canadian Biography* as its main example, so there was much code that I could ~~just cut and paste~~ carefully examine and utilise. As the examples show, it's easy to grab a webpage and analyse its content on the fly. But I wanted to process more than 10,000 pages and I knew that I was unlikely to get it working the first time round, so I decided to download the files first and then work on them locally. PH provided a basic example, to which I added some error-handling and the necessary loops to cycle through the ADB files. Because I had a bit of inside knowledge I cheated and hard-coded the numbers of articles in each volume. If I hadn't known this I would have had to scrape all the browse pages, pulling out the links and creating a list in individual ids – not hard, but a bit tedious. Anyway this is how it ended up:

```
[sourcecode lang="python"] # download_adb.py  import urllib2, time, os, sys import dh
 items = (565, 575, 607, 526, 614, 533, 543, 723, 737, 742, 737, 759, 755, 721, 703,
714, 694, 126) if os.path.exists('adb') == 0: os.mkdir('adb')  for v in range(0,18):
    for i in range (1,(items[v]+1)):         if v == 0:             filename = 'AS1%0
4db.htm' % i         else:             filename = 'A%02d%04db.htm' % (v, i)         i
f os.path.isfile('adb/' + filename) == 0:             print 'Processing: ' + filename
             url = 'http://adbonline.anu.edu.au/biogs/' + filename             try:
               response = urllib2.urlopen(url)             except IOError, e:
         if hasattr(e, 'reason'):                     print 'We failed to reach a ser
ver.'                     print 'Reason: ', e.reason                 elif hasattr(e,
'code'):                     print 'The server couldn't fulfill the request.'
             print 'Error code: ', e.code             else:                 html = re
sponse.read()                 f = open('adb/' + filename, 'w')                 f.writ
e(html)                 f.close                 time.sleep(2)         else:
   print "File already downloaded"         sys.stdout.flush()[/sourcecode]
```

Learning to count

Before too long I had a directory full of about 11,000 little html files just waiting for me to begin my evil experiments. First I had to slice them up and pull out all the interesting bits. By examining the code of the pages I could see that the main content was inside a div with the id of 'content'. Using the Beautiful Soup Python library, I was easily able to extract this div. But the content div also usually included a portrait image and a bibliography. Once again I dipped into Beautiful Soup to discard all the unwanted bits. The slicing and dicing went something like this:

```
[sourcecode language="python"]     g = open(dir + '/' + file, 'r')     html = g.read(
```

```
)     g.close()     soup = BeautifulSoup(html)     imagediv = soup.findAll(id="imageb
ox")     if len(imagediv) > 0 :         imagediv[0].extract()     heading = soup.find
All('h4')     if len(heading) > 0:         heading[0].extract()     footer = soup.fin
dAll(id="selectbib")     paras = footer[0].findNextSiblings('p')     for para in para
s:         para.extract()     footer[0].extract()     content = soup.findAll(id="cont
ent")[/sourcecode]
```

Now I had the text of the article to play with. Following the PH examples it wasn't long before I could extract word-frequency tables from a few files at a time. However, when I tried to process all the articles from a particular volume it took a verrry long time. I fiddled a bit with the code and amazed myself by dramatically improving the performance. I replaced the *wordListToFreqDict* function provided by PH with my own modified version:

```
[sourcecode language="python"] def wordListToFreqDict2(wordlist):     worddict = dict
.fromkeys(wordlist)     wordfreq = [wordlist.count(p) for p in worddict.keys()]     r
eturn dict(zip(worddict,wordfreq)) [/sourcecode]
```

The worddict = dict.fromkeys(wordlist) line made all the difference, creating a list of unique words that could then be checked against the full word list. With this hack in place I was able to process a complete volume in a few minutes.

I was already using a list of stopwords provided by PH to exclude things such as 'such' , 'as' and 'and', but obviously a few additions were necessary. To the list of stopwords I added:

```
[sourcecode language="python"] stopwords += ['january', 'february', 'march', 'april',
 'may', 'june', 'july', 'august', 'september', 'october', 'november', 'december'] sto
pwords += ['new', 'south', 'wales', 'australia', 'australian', 'victoria', 'south', '
western', 'queensland', 'tasmania'] #stopwords += ['sydney', 'melbourne', 'brisbane',
 'adelaide', 'perth', 'hobart'] stopwords += ['died', 'born', 'life', 'lived', 'marri
ed', 'father', 'wife', 'children', 'son', 'sons', 'daughter', 'daughters', 'brother',
 'brothers'] stopwords += ['street', 'st', 'year', 'years', 'months', 'acre', 'acres'
, 'ha'] stopwords += ['e', 'm', 'b', 'c', 'w', 'j', 'd', 'n', 'f', 'g', 'h', 'i', 'ii
', 'l', 'o', 'p', 'th', 'r', 't', 'u', 'r', 'nd'] [/sourcecode]
```

The first two lines should be pretty obvious. As you can see, I originally excluded names of the capital cities, but then realised that you could watch Sydney and Melbourne battle it out for pre-eminence, so I excluded the exclusion. Also out were family relations and various other words that turned up in almost every article. Cleaning out all the non-alphabetical characters from the text had left a lot of orphaned letters that had once been things like £ signs, so I had to dispose of them as well.

The modules for actually generating the clouds were mostly just copied from PH with a few minor changes. My complete script is here:

```
[sourcecode language="python"] # adb-text-count.py  import urllib2 import dh, os, sys
, time from BeautifulSoup import BeautifulSoup start = time.time() print "Started at:
 ", time.asctime(time.localtime(start)) dir = 'adb' filelist = dh.getFileNames(dir)
f = open('wordlist.txt', 'w') for file in filelist:     print 'Processing ' + file
  sys.stdout.flush()     g = open(dir + '/' + file, 'r')     html = g.read()     g.cl
```

```
ose()     soup = BeautifulSoup(html)     imagediv = soup.findAll(id="imagebox")     i
f len(imagediv) > 0 :         imagediv[0].extract()     heading = soup.findAll('h4')
    if len(heading) > 0:         heading[0].extract()     footer = soup.findAll(id="s
electbib")     paras = footer[0].findNextSiblings('p')     for para in paras:
 para.extract()     footer[0].extract()     content = soup.findAll(id="content")
text = dh.stripTags(str(content[0]))     fullwordlist = dh.stripNonAlpha(text.lower()
)     wordlist = dh.removeStopwords(fullwordlist, dh.stopwords)     f.write(" ".join(
wordlist)) f.close f = open('wordlist.txt') words = f.read() f.close wordlist = words
.split(" ") dictionary = dh.wordListToFreqDict2(wordlist) sorteddict = dh.sortFreqDic
t(dictionary) f = open('wordfreqs.txt', 'w') for s in sorteddict: f.write(str(s)+"n")
 f.close print 'Dictionary created' sys.stdout.flush() # create tag cloud and open in
 Firefox cloudsize = 200 maxfreq = sorteddict[0][0] minfreq = sorteddict[cloudsize][0
] freqrange = maxfreq - minfreq outstring = '' resorteddict = dh.reSortFreqDictAlpha(
sorteddict[:cloudsize]) print 'Creating cloud' sys.stdout.flush() for k in resorteddi
ct:     kfreq = k[0]     klabel = k[1]     klabel = dh.undecoratedHyperlink('http://a
dbonline.anu.edu.au/scripts/adbp-ent_search.php?ranktext=' + k[1] + '&search=Go!', k[
1])     scalingfactor = (kfreq - minfreq) / float(freqrange)     outstring += ' ' + d
h.scaledFontSizeSpan(klabel, scalingfactor) + ' ' dh.wrapStringInHTML("html-to-tag-cl
oud", dh.defaultCSSDiv(outstring), "Complete") finish = time.time() print "Finished a
t: ", time.asctime(time.localtime(finish)) print "Total time: ", finish - start [/sou
rcecode]
```

Biographies in 3D

To display all the portrait images in CoolIris I had to harvest all the image details and then write them to a Media RSS file for CoolIris to read.

Extracting the details of all the thumbnail versions of the portraits in the ADB was easy using Beautiful Soup. But I also need the paths to the larger versions of the portraits stored on the sites of the repositories that hold the originals. All of these sites present the images differently, so a different scraper was needed for each of them. As yet I've only included major libraries and archives – I may add some more if I get the time.

Once the paths to the thumbnails and large versions had been harvested, it was just a matter of writing the RSS feed. Actually, I created a series of RSS files, one for each volume, linked using 'rel=previous' and 'rel=next' attributes. This helped speed up the loading of the gallery. For what it's worth, the complete code is here:

```
[sourcecode language="python"] # adb-portraits.py  import socket, urllib2, urllib imp
ort dh, os, sys, time, re from BeautifulSoup import BeautifulSoup # timeout in second
s timeout = 20 socket.setdefaulttimeout(timeout) start = time.time() print "Started a
t: ", time.asctime(time.localtime(start)) dir = 'adb' for i in range(8,18):     if (i
 == 17): vol = "AS1"     else: vol = "A%02d" % i     filelist = dh.getFileNamesByVol2
(dir, vol)     f = open('adb-portraits-%s.rss' % i, 'w')     f.write("n")     f.write
("n")     f.write("n")     f.write("ADB Online Portrait Browsern")     f.write("Portr
aits of individuals included in the Australian Dictionary of Biographyn")     f.write
 ("http://www.adb.online.anu.edu.aun")     if (i > 1):         f.write ("" % (i-1))
   if (i
 0 :             print "Found an image"             sys.stdout.flush()             li
nks = imagediv[0].findAll('a')             if len(links) > 1:                 link =
urllib.unquote(links[(len(links)-1)]['href'][31:])             else:
link = urllib.unquote(links[0]['href'][31:])             print link             sys.s
tdout.flush()             try:                 response = urllib2.urlopen(link)
```

```
      except IOError, e:                if hasattr(e, 'reason'):
  print 'We failed to reach a server.'                    print 'Reason: ', e.reason
                elif hasattr(e, 'code'):                    print 'The server could
n't fulfill the request.'                    print 'Error code: ', e.code
  else:                id = str(file)[:7]                thumbnail = 'http://www.ad
b.online.anu.edu.au' + imagediv[0].img['src'].lstrip('.')                # print thu
mbnail                title = imagediv[0].p.contents[0].split(',')[0].strip().replac
e(' - ', '-')                title = title.encode('utf-8')                print "Pr
ocessing: " + title                sys.stdout.flush()                html = respons
e.read()                imgsoup = BeautifulSoup(html)                if (link.find(
'sl.nsw') > -1):                    if (link.find('ebindshow.pl') == -1): # Not thum
bnail pages - see John Bingle                        if (html.find('Higher quality i
mage') != -1):                    img = imgsoup.findAll(alt="Higher quality
image")[0].parent['href'].split('?')[1]                            #img = imgsoup.td
.a['href'].split('?')[1]                        else:                            im
g = imgsoup.table.findAll('tr')[2].img['src']                        repository = "S
tate Library of NSW"                elif (link.find('slv.vic') > -1):
      img = imgsoup.findAll(id='ImageDisplay')[0].img['src']                    repo
sitory = "State Library of Victoria"                elif (link.find('slsa.sa') > -1)
:                    img = imgsoup.findAll('td')[1].img['src']                    i
mg = link[:link.rfind('/')+1] + img                    repository = "State Library o
f SA"                elif (link.find('nla.gov') > -1):                    img = lin
k + '-v'                    repository = "National Library of Australia"
     elif (link.find('naa.gov') > -1):                    barcode = link[link.rfind(
'=')+1:]                    img = "http://naa16.naa.gov.au/rs_images/ShowImage.php?B
=%s&T=P" % barcode                        repository = "National Archives of Australia"
               elif (link.find('territorystories.nt.gov') > -1):
 img = imgsoup.table.img['src']                    repository = "Northern Territory
Library"                elif (link.find('statelibrary.tas.gov') > -1):
       if (html.find('No matches were found') == -1):                        img =im
gsoup.blockquote.img['src']                        repository = "State Library of Ta
smania"                elif (link.find('slq.qld.gov') > -1):                    img
 = imgsoup.findAll(attrs={"class":"pictureback"})[0].a['onclick']
 #img = img                    img = re.search('http://[wd/.]*.jpg', img).group()
                repository = "State Library of Queensland"                if (len(
img) > 0):                    f.write("n")                    f.write("%sn" % id)
                 f.write("%s -- %sn" % (title, repository))                    f.w
rite("http://www.adb.online.anu.edu.au/biogs/%sb.htmn" % id)                    f.wr
ite("n" % thumbnail.replace('&','&'))                    f.write("n" % img.replace('
&','&'))                    f.write("n")                    f.flush()
       print "Success!"                    sys.stdout.flush()                img =
""     f.write("n")     f.write("n")     f.close() [/sourcecode]
```

Share this:

- Click to email this to a friend (Opens in new window)
- Click to print (Opens in new window)

- Click to share on Twitter (Opens in new window)
- Share on Facebook (Opens in new window)
- Click to share on Google+ (Opens in new window)
-