Generating an organized set of links to pubmed items

VJ Carey

May 21, 2008

1 Introduction

I want to have a relatively flat but hierarchical presentation of links to published literature. My criterion for linkable publication is existence of a pubmed id; this can be made more general in the future.

An example of what I want is at `http://www.biostat.harvard.edu/~carey/provref08.html`.

To create this, I work from a hierarchical list in R. The outermost named list enumerates topics. Each element of the topic list has a named list of subtopics. Each subtopic has a vector of character pubmed IDs.

The remaining sections of this document show how to create relevant objects and methods for serialization to HTML. It was convenient to use Pau's hwriter package to do this.

2 The URL for the pubmed abstract page corresponding to a PMID

The workhorse URL generator is:

```
> options(width = 55)
> mkURL = function(id = "12584122", pre = "http://www.ncbi.nlm.nih.gov/pubmed/",
+     post = "?ordinalpos=9&itool=EntrezSystem2.PEntrez.Pubmed.Pubmed_ResultsPanel.Pu
+     paste(pre, id, post, sep = "")
+ }
```

If we execute this, we get:

```
> ur = mkURL()
> ur
```

```
[1] "http://www.ncbi.nlm.nih.gov/pubmed/12584122?ordinalpos=9&itool=EntrezSystem2.PEntr
```

This string can be passed to `browseURL` for immediate inspection, but we want to create an organized web page that includes the generated links.

3 A reference object

We propose an object structure that encodes the organization.

```
> setClass("litRef", representation(topic = "character",
+     subtopic = "character", pmid = "character"))

[1] "litRef"

> mkLitRef = function(topic, subtopic, pmid) {
+     new("litRef", topic = topic, subtopic = subtopic,
+         pmid = pmid)
+ }
> setMethod("show", "litRef", function(object) {
+     cat("pubmed ref: pmid ", object@pmid,
+         "\n")
+     cat("topic: ", object@topic, "\n")
+ })

[1] "show"

> ex = mkLitRef("Multiple testing", "FDR/microarray",
+     "12584122")
> ex

pubmed ref: pmid  12584122
topic:  Multiple testing
```

That's perhaps a minimally sufficient schema. Now we want to use the web to populate some additional fields to make the report useful.

4 Using the pubmed() XML response to get key fields

First we define a function that extracts key information from a pubmed xml query:

```
> pmidKeyData = function(pmid = "12584122",
+     maxauth = 3) {
+     require("XML") || stop("need the XML package for this function")
+     x = pubmed(pmid)
+     rr = xmlRoot(x)
+     top = xmlChildren(rr)
+     pmart = top[["PubmedArticle"]]
+     cit = xmlChildren(pmart)[["MedlineCitation"]]
+     art = cit[["Article"]]
+     cart = xmlChildren(art)
+     yr = xmlValue(xmlChildren(xmlChildren(xmlChildren(cart[["Journal"]])$JournalIss
+     tmp = lapply(xmlChildren(cart$AuthorList),
+         xmlChildren)
+     al = sapply(tmp, function(x) xmlValue(x$LastName))
+     if (length(al) > 3)
+         al = c(al[1:3], "et al.")
+     journalTitle = try(xmlValue(xmlChildren(cart$Journal)$ISOAbbreviation),
+         silent = TRUE)
+     if (inherits(journalTitle, "try-error"))
+         journalTitle = try(xmlValue(xmlChildren(cart$Journal)$Title))
+     list(journalTitle = journalTitle, articleTitle = xmlValue(cart$ArticleTitle),
+         authorList = paste(al, collapse = ", "),
+         year = yr)
+ }
```

5 A richer 'entry' object

Now we use this to populate the enriched entry:

```
> setClass("litEntry", representation(auth = "character",
+     title = "character", journal = "character",
+     year = "character"), contains = "litRef")

[1] "litEntry"

> setGeneric("makeEntry", function(lr) standardGeneric("makeEntry"))

[1] "makeEntry"

> setMethod("makeEntry", "litRef", function(lr) {
+     require(annotate)
+     x = pmidKeyData(lr@pmid)
+     new("litEntry", auth = x$authorList, title = x$articleTitle,
+         journal = x$journalTitle, year = x$year,
+         lr)
+ })

[1] "makeEntry"
```

```
> setMethod("show", "litEntry", function(object) {
+     cat("literature entry:\n")
+     callNextMethod()
+     cat("  title: ", object@title)
+     cat("\n")
+     cat("  journal: ", object@journal, "(",
+         object@year, ")\n")
+ })
```

```
[1] "show"
```

```
> ddd = makeEntry(ex)
> ddd
```

```
literature entry:
pubmed ref: pmid  12584122
topic:  Multiple testing
  title:  Identifying differentially expressed genes using false discovery rate control
  journal:  Bioinformatics ( 2003 )
```

6 An example schema (list) for a page

To specify the page to be generated, we use a hierarchical list.

```
> mylit = list()
> topics = c("Cautions", "Yeast", "Expression arrays",
+     "Expression plus genotyping", "Copy number variation",
+     "Statistics", "Computing")
> for (i in 1:length(topics)) mylit[[topics[i]]] = list()
> mylit[["Cautions"]][["Irreproducibility"]] = c("17987014",
+     "15814023")
> mylit[["Cautions"]][["Batch effects"]] = c("17597765",
+     "18309960")
> mylit[["Yeast"]][["Cell cycle"]] = c("9843569",
+     "12399584")
> mylit[["Yeast"]][["Environmental stress response"]] = c("11102521")
> mylit[["Expression arrays"]][["Breast cancer/stem compendium"]] = c("18443585")
> mylit[["Expression arrays"]][["Pancreatic cancer/pathway addiction"]] = c("18413752
> mylit[["Expression plus genotyping"]][["Early work"]] = c("16251966",
+     "17158513")
> mylit[["Expression plus genotyping"]][["Multiple populations"]] = c("17206142",
+     "17873874")
```

```
> mylit[["Expression plus genotyping"]][["Drosophila"]] = c("17873888",
+     "17418441")
> mylit[["Copy number variation"]][["Breast cancer cell lines"]] = c("17157791")
> mylit[["Copy number variation"]][["Glioma"]] = c("18077431")
> mylit[["Statistics"]][["Linear models"]] = c("16646809")
> mylit[["Statistics"]][["Multiple testing"]] = c("12584122")
> mylit[["Statistics"]][["GSEA"]] = c("16199517",
+     "17903287", "17127676")
> mylit[["Statistics"]][["Regularized LDA"]] = c("16603682")
> mylit[["Computing"]][["Bioconductor"]] = c("15461798")
> mylit[["Computing"]][["Reproducible research"]] = c("16646837")
```

7 Walking the schema to generate the page

We get a list of topics and subtopics as follows:

```
> tops = lapply(mylit, names)
> topics = names(tops)
```

We need to walk through this and create a list of litRef instances:

```
> reflist = list()
> for (i in 1:length(tops)) {
+     reflist[[topics[i]]] = list()
+     for (j in 1:length(tops[[i]])) reflist[[topics[i]]][[tops[[i]][j]]] = lapply(my
+         function(x) mkLitRef(topics[i], tops[[i]][j],
+             x))
+ }
> entlist = lapply(reflist, lapply, lapply,
+     makeEntry)
```

Now we need to be able to serialize entries to HTML. We use the hwriter package.

```
> setGeneric("HW", function(x, con) standardGeneric("HW"))

[1] "HW"

> setMethod("HW", c("litEntry", "ANY"), function(x,
+     con) {
+     hwrite(x@title, con, link = mkURL(id = x@pmid),
+         style = "padding-left:30pt; font-family:sans-serif",
+         br = TRUE)
+     hwrite(x@auth, con, style = "padding-left: 40pt; font-family:sans-serif")
+     hwrite(", ", con)
+     hwrite(x@journal, con, style = "font-weight: bold; font-family:sans-serif")
+     hwrite("(", con)
+     hwrite(x@year, con, style = "font-family:sans-serif")
+     hwrite(")", con, br = TRUE)
+ })
```

```
[1] "HW"
```

The whole hierarchical structure is then serialized to HTML using hwriter.

```
> library(hwriter)
> kk = openPage("demowrite.html", ".")
> hwrite("A provisional set of references for preparation for CDATA-08",
+     kk, style = "font-weight: bold; font-size:130%; font-family:sans-serif",
+     br = TRUE)
> hwrite(" ", kk, br = TRUE)
> tl = lapply(entlist, names)
> for (i in 1:length(tl)) {
+     hwrite(names(tl)[i], kk, style = "font-weight: bold; font-size: 130%; font-fami
+         br = TRUE)
+     for (j in 1:length(tl[[i]])) {
+         hwrite(tl[[i]][j], kk, style = "padding-left:20pt; font-size: 115%; font-fa
+             br = TRUE)
+         for (k in 1:length(entlist[[i]][[j]])) HW(entlist[[i]][[j]][[k]],
+             kk)
+     }
+ }
> closePage(kk)
```

8 Things to do

- generalize to other kinds of links, such as doi or direct URL specs ... the problem here is that we may not get publishing metadata so easily and we probably have to hand code author info, etc., ... but semantic web should eventually solve that
- meaningful links to software resources
- combine with task view system