Challenges in automated DOI resolution

Martin Fenner

October 13, 2013

Yesterday we created a set of roughly 10,000 DOIs for journal articles published in 2011 or 2012. We used these DOIs as a reference set in a data hackathon around article-level metrics/altmetrics - material for another blog post.

The random DOIs were generated using the CrossRef RanDOIm service, with article titles fetched from the CrossRef OpenURL API. We didn't have time to properly parse the publication date and only used the publication year. We used the `crossref_r` and `crossref` functions from the rOpenSci rplos package (and some extra help from Scott Chamberlain) to achieve this, the datasets were deposited to figshare and can be found here (2011) and here (2012).

The basic idea behind DOI names is summarized well in the Wikipedia entry:

> A digital object identifier (DOI) is a character string (a "digital identifier") used to uniquely identify an object such as an electronic document. Metadata about the object is stored in association with the DOI name and this metadata may include a location, such as a URL, where the object can be found. The DOI for a document is permanent, whereas its location and other metadata may change. Referring to an online document by its DOI provides more stable linking than simply referring to it by its URL, because if its URL changes, the publisher need only update the metadata for the DOI to link to the new URL.

DOIs for journal articles should provide users with a URL specific for that journal article. This URL could point to a digital copy of the journal article in HTML or PDF format, or could point to a landing page (with an abstract or other basic metadata) for journal articles that require a subscription. This should work not only for humans using a web browser, but also for automated services using command line tools such as curl as scientific infrastructure depends heavily on automation and computers talking to each other. In our use case we want to find content linking to a specific article, and as some services (e.g. social media) will use the URL and not DOI of an article, we need to find out that URL.

Unfortunately it was difficult to find a URL for many DOIs in our reference set using automated tools. All these DOIs resolve to URLs for human users using a

web browser, but for automated tools there are a number of challenges:

Requiring a cookie

Some publishers require a cookie, and that can cause problems for automated tools. We can use the popular command line tool `curl` with the options `-L` to follow redirects and `-I` to only send the header (as we care about the location and not the content of the page).

```
curl -I -L "http://dx.doi.org/10.1080/13658816.2010.531020"
```

This command will lead us not to a page specific for that article, but to a "Cookie absent" page. You can work around this by having curl accept cookies:

```
curl -I -L --cookie "tmp" "http://dx.doi.org/10.1080/13658816.2010.531020"
```

Unfortunately not all tools do this. The way Facebook tracks likes, shares, comments, etc. is a prominent example.

Too many redirects

Some DOIs never resolve using a HEAD request, and curl stops after 50 redirects:

```
curl -I -L "http://dx.doi.org/10.1097/SLA.0b013e318235e525"
```

This error may relate to the "requiring a cookie" error above.

Method not allowed

Some DOis HEAD requests result in a "405 Method Not Allowed" error. The reason is that the journal platform doesn't accept the HEAD request, but wants a GET instead.

```
curl -I -L "http://dx.doi.org/10.1002/sam.10120"
```

The HTTP 1.1 protocol says about HEAD:

> The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response. ... This method is often used for testing hypertext links for validity, accessibility, and recent modification.

We can work around this error by using a GET request, which unfortunately creates extra overhead and is not the recommended way to obtain this kind of information.

Empty reply from server

Some DOIs never resolve using a HEAD because curl reports "Empty reply from server" and we don't get a HTTP 200 status code.

```
curl -I -L "http://dx.doi.org/10.1016/j.cca.2011.04.012"
```

You can again work around this by using the location information before the last redirect, but maybe resolving a DOI should not result in curl routinely throwing an error. It looks as if this error is related to "method not allowed", as a GET request resolves to a landing page.

This problem is not specific to the `curl` tool, we get exactly the same error with `wget`:

```
wget -S --spider "http://dx.doi.org/10.1016/j.cca.2011.04.012"
```

Timeout errors

Some DOI resolutions resulted in timeout errors, but this was temporary and much less frequent than the errors above.

Resource not found

We didn't specifically look into this error, which is a well-known problem with URLs. The DOI names we used were from 2011 and 2012, and it is known that link rot is more common the older the resource is.

Content negotiation

As Karl Ward has pointed out in the comments there are other ways to get to the URL from the DOI name, e.g. using content negotiation:

```
curl -LH "Accept: application/vnd.crossref.unixref+xml" "http://dx.doi.org/10.1016/j.cca.201
```

The URL is stored in the `doi_data/resource` attribute. The URL stored there is unfortunately not always the final landing page for the article, e.g. for the DOI name used in the example above.

Conclusions

We created a reference set of 10,000 DOIs to collect metrics around them. The first conclusion from this exercise is that getting the URL for these articles is a challenge in many cases. This does not seem to relate to a permission problem for subscription content, but rather how the HTTP HEAD request is handled. Content negotiation is one alternative, but sometimes leads to different URLs for the landing page than where the user would get via the browser. We therefore have to rewrite our code to use GET requests and to better handle the scenarios above.

Update 10/13/13: Updated the title and the text to make it clear that I am not talking about DOIs that don't resolve for human users, but rather about the problems automating this process using command-line tools.