Proposal to NLNet: O'Search - Offline Search

Sean Levy

14-JUL-2015

Offline Search

The most private search is the one that never hits the wire: what if you could plug cheap, external storage (say, a USB stick) into your device of choice and search a credible, usable catalog of the web without a packet ever being sent or received? No logs for TLAs to pick over *post-hoc*, no censorship to deal with other than the explicit editorial policy used to build the catalog.

The overall vision in the medium term is to produce a catalog of the web that fits in something like 1TB and could be carried around (eventually on a USB stick), shared via Sneaker-Net and dead drops for those without 1st world connectivity and spread via Bit-Torrent by those who do. The catalog would have to be updated regularly; cranking out a new version every 30 days would be a reasonable goal to start with, the idea being that the number of days could be reduced if more resources were available. The canonical users would be journalists and activists, rural people and others with limited or no connectivity, privacy advocates and communities who wish to have their own view of the web for whatever reasons (religious, cultural, political).

Some of these prospective user communities might be interested in cooking their own catalogs, for whatever reason. A twin goal, therefore, is to release the software used to build the catalog under the ISC license[1] (a simplified version of the BSD license[2] favored by some projects such as OpenBSD[3]). The software is designed so that groups who are interested in producing catalogs can collaborate so long as they agree on certain basic parameters. The more the merrier: let a thousand flowers bloom. If you have the resources to crawl the web you can run the whole stack yourself. If you don't want to do the crawling but are interested in tweaking a catalog in some other way you can subscribe to another group's crawl and build your own catalog. There are good reasons why you'd want to do both of those things. Of course, subscribing to someone else's crawl presumes that you already agree with enough of that crawl's parameters. For instance, a group of religious fundamentalists who wanted to make a catalog that does not contain any material antagonistic to their beliefs would probably not want to subscribe to a crawl produced by a group of hacktivists.

The software will be as minimal as possible, creating anew only what is missing from the open-source world. I have identified several major components of what is necessary and think I have a fairly good handle on what is missing for such a system and what the path forward looks like to the first catalog release.

The catalog will come with a reference implementation of a search tool, command-line only to start with, written in Python, Perl or some other language something that could be

[1] https://en.wikipedia.org/wiki/ISC_license
[2] https://en.wikipedia.org/wiki/BSD_license
[3] http://www.openbsd.org

packaged up in a universally consumable way. More complex, user-friendly tools to make use of the catalog could be written by anyone. I intend to use Xapian[4] for the indexing back end, and the files we would distribute would be in the format that the Xapian client libraries expect. This means any programming language with Xapian bindings could be used to build search tools. Xapian's architecture allows for custom back-ends to be built; if it becomes necessary in order to reduce the size of the catalog we can do whatever we need to do without sacrificing the rest of Xapian's features, such as its extensible query language.

Editorial Policies for Catalogs

The specific choice of 1TB is arbitrary: it's a previously unimaginable amount of storage which will shortly become commonplace. The terabyte is the new gigabyte, just like always. The actual size is more or less irrelevant because I'm talking about editing (a catalog of) the web to make it more manageable, so fundamentally the idea is to trade time for space. There hasn't been too much attention paid to the idea of minimizing *space* in the search and indexing world - mostly everyone wants to minimize *time* and is willing to trade space for time, especially given how cheap storage has become relative to other resources.

Our paradigm is different: we instead wish to put an upper bound on the size of the catalog and will trade both run-time (both in the catalog build and in the end user experience) when possible to shrink it. This decision colors everything else in many ways.

Whatever the specific maximum size ends up being, it is still obvious that *something* will have to be left out of The Whole Enchilada. If we are to believe pundits the web's ever-expanding hugeness will continue on an exponential arc until the heat death of the universe. Although it is true that the total size of the web in several dimensions continues to grow, it is not true that everyone wants to search all of it all of the time.

As a straw-man I propose the following *editorial policy* for a catalog of the web that might fit in 1TB:

Drop the following:

1. Porn;
2. Social Media (Anything on Facebook, LinkedIn, G+, ...);
3. Content in anything but a single language (in this case English);
4. SEO Garbage (pages that only exist to game PageRank).

It might be that some prospective user communities won't agree with my assessment, not just of Facebook, but in general: if you're a journalist researching a piece on porn then you do in fact want to find porn (not to be overly puritanical). This means that at least the catalog I would like to produce use will not necessarily be suitable for everyone. This is why the editorial machinery used to drop things from the catalog is not fixed: in fact this is the most important piece of the puzzle that does not yet exist.

Nonetheless my short kill file[5]-style policy is just a sketch: without some numbers I don't really know if the goal of 1TB is achievable this way. This is why I propose to first do a pre-catalog survey of the web in order to produce numbers that can be used to do what-if analyses like:

[4] http://xapian.org
[5] https://en.wikipedia.org/wiki/Kill_file

- How big is porn in its various dimensions (urls, bytes, stems)?
- What is the effect on the final size of the Xapian files if we drop it?

Answering the latter question requires a little modeling and experimentation but the idea should be clear: there might be very "large" areas of the web that don't contribute proportionally to the size of the index. I should have enough resources with the hardware available to me to store meta-data about enough of the web to make this possible... nothing like enough space to actually store the whole web, of course, but enough to make reasonable estimates about size as it impacts the catalog.

Once the survey is done the next step would be to come up with a final size constraint that is sensible given *(a)* the state of portable media at the time and *(b)* what we've learned from the survey.

In doing the survey it is certain that issues will arise that will require code to be written, design assumptions to be revisited, etc. For this reason I think it only makes sense to think about the architecture in general terms and not get too specific too early. My overall vision for the catalog builder is analogous to a snake that eats its own tail; intermediate results and content from the web can only be partially cached due to size constraints, so estimating the high watermark for temporary storage during a build is crucial. This is as opposed to a system such as Google's, which is more like a large herd of goats romping around in an effectively infinite field, a.k.a. their cache of the web. We can improve performance by running multiple *ouroboros* instances but Google's herd of goats is always going to win on speed.

Search Interfaces and Caching

Our overriding concern is privacy; the specific catalog we wish to produce is one geared towards activists, journalists and others on the front line of the war against privacy being waged by the largest, most aggressive governments on the planet. To this end we also wish to explore a few other ideas in the user interface. Although this is a secondary area we will be providing a reference tool with the catalog that can be used by itself to search, and which provides us with a platform to experiment with privacy-related features in this context.

Users of most online search engines have gotten used to a slew of features that can easily violate their privacy or otherwise leak information about them. Search results pages frequently contain bugs of many kinds; they also include summaries and extracts from web pages that appear in them, etc. Our catalogs, being limited in space, will surely not contain enough information by themselves to mimic these features without code to reach out and pre-fetch results to produce these kinds of summaries, snapshots, etc.

Any instance of our catalog will also have an associated cache, which will normally ship as an empty folder. As a user interacts with the catalog, subject to user preferences, this cache could accrue up-to-date summary information on results as they come out of searches, so that over time more user-friendly results could be displayed if cached information were available or if the user agreed to allow on-the-fly network access from the tool being used to search the catalog.

We propose that the reference client have three modes:

1. Catalog only: only information that appears in the catalog is displayed, which will result in fairly minimalist information for search hits. The cache is ignored in this mode, no summaries or snapshots are available;

2. Catalog + Cache: information in the cache that is not deemed too old (user settable) will be merged into search results, so that some hits might have more elaborate summaries, images, etc. available;

3. Full Cache: the cache is filled on the fly for all search hits that are displayed. Cache entries that are too old are refreshed.

In the first two modes no network traffic is generated by searching; only in the last mode will information about what the user is searching for leak onto the wire.

A journalist who was preparing to go into a war-torn area where getting online is to be avoided if possible could use the Full Cache mode to pre-load their cache over Tor by running a set of queries that loads the cache with summaries and snapshots for all search hits, before they go. They could then use their catalog with its associated cache to see reasonably rich results (assuming the cache covered them), and which allowed them to avoid using the network as much as possible while in inclement circumstances.

Proposal

This is an ambitious project on the whole but a substantial portion of it is doable in a single year by a motivated, qualified hacker with appropriate support, especially given some of the excellent open-source software that already exists that can be brought to bear on the problem. I am asking for USD$30k to support my work for a year, during which I will build enough of the system to complete the initial survey of the web, come up with a target size and produce the first catalog and associated software. It may be that all of the features of the reference search tool are not implemented and that the editorial policy machinery does not support everything necessary for every potential user to build the catalog they desire in this time-frame. Some effort must also be spent on building a community of interested developers and early users, as well as feeding back relevant developments and fixes to open-source projects we use. Sites like GitHub are ideal for this and also serve as a dissemination mechanism.

In addition to the money I am also looking for other kinds of support that an organization like NLnet might be uniquely suited to provide. One resource that this effort obviously requires is bandwidth for spidering the web. NLnet might be able to connect this project with entities that would be interested in donating bandwidth and other resources to a project such as this, for instance.