COSCS494/594 Fundamentals of Digital Archeology
Data Discovery

Audris Mockus
University of Tennessee
audris@utk.edu

Where can I find data? (early history)

Early history

- Cultural artifacts (people's mind): 3M-5K years ago
  - Passed on verbally, through practice and joint experiences
- Physical Artifacts
  - Stone/clay tablets/papyrus/paper: 5K-now

Where can I find data? (Digital Age)

Digital Age

- Mainframe era
  - 1928 Magnetic tape
  - 1932 Magnetic Drum
  - 1956 Hard disk
- Minicomputers/PCs
  - 1980 CD, 1995 DVD
  - 1990 PCs
- 2010 Cloud era
  - SourceForge
  - SalesForce

Where can I find data? (Now)

In the cloud

- Public internet
- Corporate internet
- In what form?
  - Web pages
  - Within applications (Deep Web)

What is Deep Web

- Traditional Web Spider/Crawler
  - Start from a set of URLs
  - Retrieve and extract all the links
  - Snowball
    - follow all the new links until no more found
  - Periodically revisit all the URLs found so far to see if there are new links
- Deep Web: most of the content you can not find using the method above
  - Search: Google/BitBucket/SourceForge/GoogleCode
  - API's (typically REST), e.g, github, Twitter, Facebook
  - Other interfaces:
    - ITS: Bugzilla, Debian, Ubuntu, JIRA, ...
  - Application commands for VCS: hg, git, svn, bazaar, CVS
  - POST (web forms) interface to relational databases

REST APIs

E.g. GitHub

`https://developer.github.com/v3/`

- For example, see:

`https://api.github.com/repos/fdac/syllabus/issues?state=closed`

- The output is json (as your iPython notebook)

```
 [
  {
    "url": "https://api.github.com/repos/fdac/syllabus/issu
    "labels_url": "https://api.github.com/repos/fdac/syllab
    "comments_url": "https://api.github.com/repos/fdac/syll
    "events_url": "https://api.github.com/repos/fdac/syllab
...
```

Search

E.g., GoogleCode

`http://code.google.com/hosting/search?filter=0&q=label:SEARCHSTR&start=0`

- Page through responses
- Extract relevant links

Application commands

Extract data from VCS: list revisions

- CVS: cvs log PRJ
- SVN: svn log -v –non-interactive PRJ
- Mercurial: hg log -v PRJ
- Bazaar: bzr log -v $–$long PRJ
- GIT: git –git-dir=PRJ log –numstat -M -C –diff-filter=ACMR –full-history –pretty=tformat:"%n%H;%T;%P;%an;%ae;%at;%cn;%ce;%ct;%s"

Discovery Ethics (Don't get banned)

Don't get banned

- Check if the site has relevant policies, e.g.,
  - robots.txt
  - Text in http responses, e.g, `https://bugzilla.gnome.org/show_bug.cgi?id=309324`
  - Rate limiting, e.g `https://dev.twitter.com/docs/rate-limiting-faq`
- Be sensitive of potential harm you may cause
  - Bringing the server down
  - Exposing information that was not intended to be public
- Be aware of context:
  - Don't treat Google as a small project
  - Don't treat a small project as Facebook

Discovery Ethics (Workarounds)

Workarounds

- Ask for dump (e.g., Mozilla Bugzilla)
- Use authentication
  - Typically increasess allowed rate
  - Sometimes provides more detail (Gnome Bugzilla)
- Create multiple accounts
- Run from multiple ip addresses
- Keep the rate below threshold

Discovery Ethics (do it once)

- Retrieve once and save

```
import pickle
r = requests .get(URL)
# to save
pickle.dump (r, open('storedReq.obj', 'w'))
# to restore
rLater = pickle.load(open('storedReq.obj', 'r'))
```

A few legal aspects

- Be careful with spiders
  - They may go to sites that you would not consider visiting
- If collecting competitive intelligence, there are legal restrictions on what you could use: make sure the obtained information is, indeed, public
- If doing a research study, you may need to pass the plan through IRRB
- There may be privacy issues:
  - Anonymize whenever possible
  - Report aggregates

A few references

- Way back machine: `http://archive.org/web/`
- `http://en.wikipedia.org/wiki/Electronic_discovery`
- Audris Mockus Amassing and indexing a large sample of version control systems: towards the census of public source code history In 6th IEEE Working Conference on Mining Software Repositories, May 16-17 2009. `http://mockus.org/papers/amassing-slides.pdf`
- Audris Mockus Large-scale code reuse in open source software. In ICSE'07 Intl. Workshop on Emerging Trends in FLOSS Research and Development, Minneapolis, Minnesota, May 21 2007. `http://mockus.org/papers/ossreuse.slides.pdf`

A crawler

```
while (list of unvisited URLs is not empty):
 take URL from list
 fetch content
 record the content if desired
 if content is HTML:
  parse out URLs from links
  foreach URL:
   if (it matches your rules and
      it's not already in either
      the visited or unvisited list):
    add it to the unvisited list
```