Large Crawls of the Web for Linguistic Purposes

Marco Baroni

SSLMIT, University of Bologna

Birmingham, July 2005
The WaCky approach

- http://wacky.sslmit.unibo.it
- Use existing open tools, make developed tools publicly available.
- Please join us (for other languages as well!)
The basic steps

- Select “seed” urls.
- Crawl.
- Post-processing.
- Linguistic annotation.
- Indexing, etc.
Selecting seed urls

- Use queries for random word combinations to Google search engine.
Selecting seed urls

- Use queries for random word combinations to Google search engine.
- Start crawl from urls discovered in this way.
Selecting seed urls

- Use queries for random word combinations to Google search engine.
- Start crawl from urls discovered in this way.
- Which random words?
  - Middle-frequency words from general/newspaper corpus ("public").
  - Basic vocabulary list ("private").
Selecting seed urls

- Use queries for random word combinations to Google search engine.
- Start crawl from urls discovered in this way.
- Which random words?
  - Middle-frequency words from general/newspaper corpus (“public”).
  - Basic vocabulary list (“private”).
- How random are the urls collected in this way? Ongoing work with Massimiliano Ciaramita (ISTC, Rome).
Outline

1. Introduction
2. Selecting seed urls
3. **Crawling**
   - Basics
   - Heritrix
   - My ongoing crawl
4. Post-processing
   - Filtering and cleaning
   - Language identification
   - Near-duplicate spotting
5. Conclusion
   - Annotation
   - Indexing, etc.
   - Summing up and open issues
Crawling

- Fetch pages, extract links.
- Follow links, fetch pages.
Important in a good crawler

- Honoring robots.txt, politeness
Important in a good crawler

- Honoring robots.txt, politeness
- Efficiency, multi-threading, robust “Frontier”
Important in a good crawler

- Honoring robots.txt, politeness
- Efficiency, multi-threading, robust “Frontier”
- Avoid spider traps
Important in a good crawler

- Honoring robots.txt, politeness
- Efficiency, multi-threading, robust “Frontier”
- Avoid spider traps
- Control over crawl scope
Important in a good crawler

- Honoring robots.txt, politeness
- Efficiency, multi-threading, robust “Frontier”
- Avoid spider traps
- Control over crawl scope
- Progress monitoring
Important in a good crawler

- Honoring robots.txt, politeness
- Efficiency, multi-threading, robust “Frontier”
- Avoid spider traps
- Control over crawl scope
- Progress monitoring
- Intelligent management of downloaded text
Important in a good crawler

- Honoring robots.txt, politeness
- Efficiency, multi-threading, robust “Frontier”
- Avoid spider traps
- Control over crawl scope
- Progress monitoring
- Intelligent management of downloaded text
- Works out of the box, reasonable defaults
<table>
<thead>
<tr>
<th>Basics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heritrix</td>
</tr>
</tbody>
</table>

Heritrix

http://crawler.archive.org/
Heritrix

- http://crawler.archive.org/
- Free/open crawler of Internet Archive
Heritrix

- http://crawler.archive.org/
- Free/open crawler of Internet Archive
- Very active, supporting community…
Heritrix

- Free/open crawler of Internet Archive
- Very active, supporting community…
- that includes linguists and machine learning experts
### The Heritrix WUI

**Status of crawler as of Jul. 11, 2005 11:11:33 GMT**  
Alerts: no alerts

Crawler is running
Current job: secdelarger

0 jobs pending, 2 completed  
Downloaded 9376524 documents in 99 h., 45 min. and 0 sec.

**Crawler status**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crawler running:</td>
<td>Yes</td>
</tr>
<tr>
<td>Current job:</td>
<td>secdelarger</td>
</tr>
<tr>
<td>Jobs pending:</td>
<td>0</td>
</tr>
<tr>
<td>Jobs completed:</td>
<td>2</td>
</tr>
<tr>
<td>Status:</td>
<td>Running</td>
</tr>
<tr>
<td>Processed docs/sec:</td>
<td>17.7 (26.11) KB/sec</td>
</tr>
<tr>
<td>Run time:</td>
<td>99 h., 45 min. and 0 sec.</td>
</tr>
</tbody>
</table>

**Used memory:** 1463199 KB  
**Heap size:** 2081024 KB  
**Max heap size:** 2081024 KB

**Alerts:** 0 (0 new)  
**Active thread count:** 145 of 149  
**Total data received:** 185 GB

![Heritrix WUI](image)

Stop crawling pending jobs | Terminate current job | Pause current job | Refresh

Shut down Heritrix software | Logout
The output of Heritrix

- Documents distributed across gzipped “arc” files not larger than 100 MB.
- Info about retrieved docs (fingerprints, size, path) in arc file headers and in log files.
My German crawl

- On server running RH Fedora Core 3 with 4 GB RAM, Dual Xeon 4.3 GHz CPUs, about 1.1 TB hard disk space.
- Seeded from random Google queries for SDZ and basic vocabulary list terms.
- 8631 urls, all from different domains.
- SURT scope:
  - `http://at`,
  - `http://de`,
- Tom Emerson’s regexp to “focus on HTML”
- For most settings, Heritrix defaults.
Current status of crawl

- In about a week:
- Retrieved about 265 GB, about 54 GB of arc files
- In earlier experiments, 7 GB arc files yielded about 250M words after cleaning.
Outline

1. Introduction
2. Selecting seed urls
3. Crawling
   - Basics
   - Heritrix
   - My ongoing crawl
4. Post-processing
   - Filtering and cleaning
   - Language identification
   - Near-duplicate spotting
5. Conclusion
   - Annotation
   - Indexing, etc.
   - Summing up and open issues
Post-processing

- Various forms of filtering, boilerplate stripping
Post-processing

- Various forms of filtering, boilerplate stripping
- Language identification
Post-processing

- Various forms of filtering, boilerplate stripping
- Language identification
- Near-duplicate identification
Filtering as you crawl...

Wouldn’t it be nice to filter as you crawl?
Filtering as you crawl...

Wouldn’t it be nice to filter as you crawl?

Yes, but:
Filtering as you crawl . . .

Wouldn’t it be nice to filter as you crawl?
Yes, but:
  - You don’t know what you’ve got until you download it
Filtering as you crawl...

- Wouldn’t it be nice to filter as you crawl?
- Yes, but:
  - You don’t know what you’ve got until you download it
  - Some pages are “bad” for corpus, but “good” for crawling
Filtering as you crawl...

- Wouldn’t it be nice to filter as you crawl?
- Yes, but:
  - You don’t know what you’ve got until you download it
  - Some pages are “bad” for corpus, but “good” for crawling
- Promising: brand new Heritrix/Rainbow interface.
Filters and boilerplate removal

- Ignore docs smaller than 5KB, larger than 200KB.
Filters and boilerplate removal

- Ignore docs smaller than 5KB, larger than 200KB.
- Porn stop words (not out of prudery, but because pornographers do funny things with language to fool search engines).
Filters and boilerplate removal

- Ignore docs smaller than 5KB, larger than 200KB.
- Porn stop words (not out of prudery, but because pornographers do funny things with language to fool search engines).
- Boilerplate removal: see next talk.
Language identification

- *After* boilerplate removal.
Language identification

- After boilerplate removal.
- Among the options:
  - Van Noord’s TextCat tool: Not robust (not German if nouns not in uppercase).
  - Efficiency problems?
  - Small list of function words: In my experiments, fast and effective.
  - Minimum proportion of function words also good to detect connected prose (Zipf to our rescue).
  - Non-latin1 languages: recognize language and encoding.
Language identification

- *After* boilerplate removal.
- Among the options:
  - Van Noord’s TextCat tool:
    - Not robust (not German if nouns not in uppercase).
    - Efficiency problems?
After boilerplate removal.

Among the options:

- Van Noord’s TextCat tool:
  - Not robust (not German if nouns not in uppercase).
  - Efficiency problems?

- Small list of function words:
  - In my experiments, fast and effective.
  - Minimum proportion of function words also good to detect connected prose (Zipf to our rescue).
After boilerplate removal.
Among the options:
- Van Noord’s TextCat tool:
  - Not robust (not German if nouns not in uppercase).
  - Efficiency problems?
- Small list of function words:
  - In my experiments, fast and effective.
  - Minimum proportion of function words also good to detect connected prose (Zipf to our rescue).
- Non-latin1 languages: recognize language and encoding
Near-duplicate spotting


- Freely available implementation in perl and MySQL written with Eros Zanchetta (SSLMIT).
The shingling algorithm

- For each page, randomly sample $N$ n-grams (e.g., 25 pentagrams)
- Look for pages that share at least $X$ of the randomly sampled n-grams (e.g., 5)
- (Important to do boilerplate removal before, or most of your n-grams will be things like: “buy click here”.)
What are near-duplicates, exactly?

- Once boilerplate and small docs are removed, not that many near-duplicates.
What are near-duplicates, exactly?

- Once boilerplate and small docs are removed, not that many near-duplicates.
- Should we really be throwing them away?
Outline

1. Introduction
2. Selecting seed urls
3. Crawling
   - Basics
   - Heritrix
   - My ongoing crawl
4. Post-processing
   - Filtering and cleaning
   - Language identification
   - Near-duplicate spotting
5. Conclusion
   - Annotation
   - Indexing, etc.
   - Summing up and open issues
With standard tools... 
However, need for robustness. 
Following wreaks havoc on TreeTagger tokenizer and tagger:
und bewusst werden. ein unsichtbares band verbindet
Indexing, retrieval, interfaces...

- CWB, SketchEngine, Xaira?
- Lucene?
- MySQL?
Building a large corpus by crawling is quite straightforward...
Conclusion

- Building a large corpus by crawling is quite straightforward...
- but devil is in the (terabytes of) details.
Conclusion

Building a large corpus by crawling is quite straightforward. . .

but devil is in the (terabytes of) details.

Some (of many) open issues:

- What “language” are we sampling from?
- How large is large enough?