Large Crawls of the Web for Linguistic Purposes

Marco Baroni

SSLMIT, University of Bologna

Birmingham, July 2005

Outline

The WaCky approach

- `http://wacky.sslmit.unibo.it`
- Current target: 1-billion token English, German, Italian Web-corpora by 2006.
- 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.

Outline

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

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

Heritrix

- `http://crawler.archive.org/`

Heritrix

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

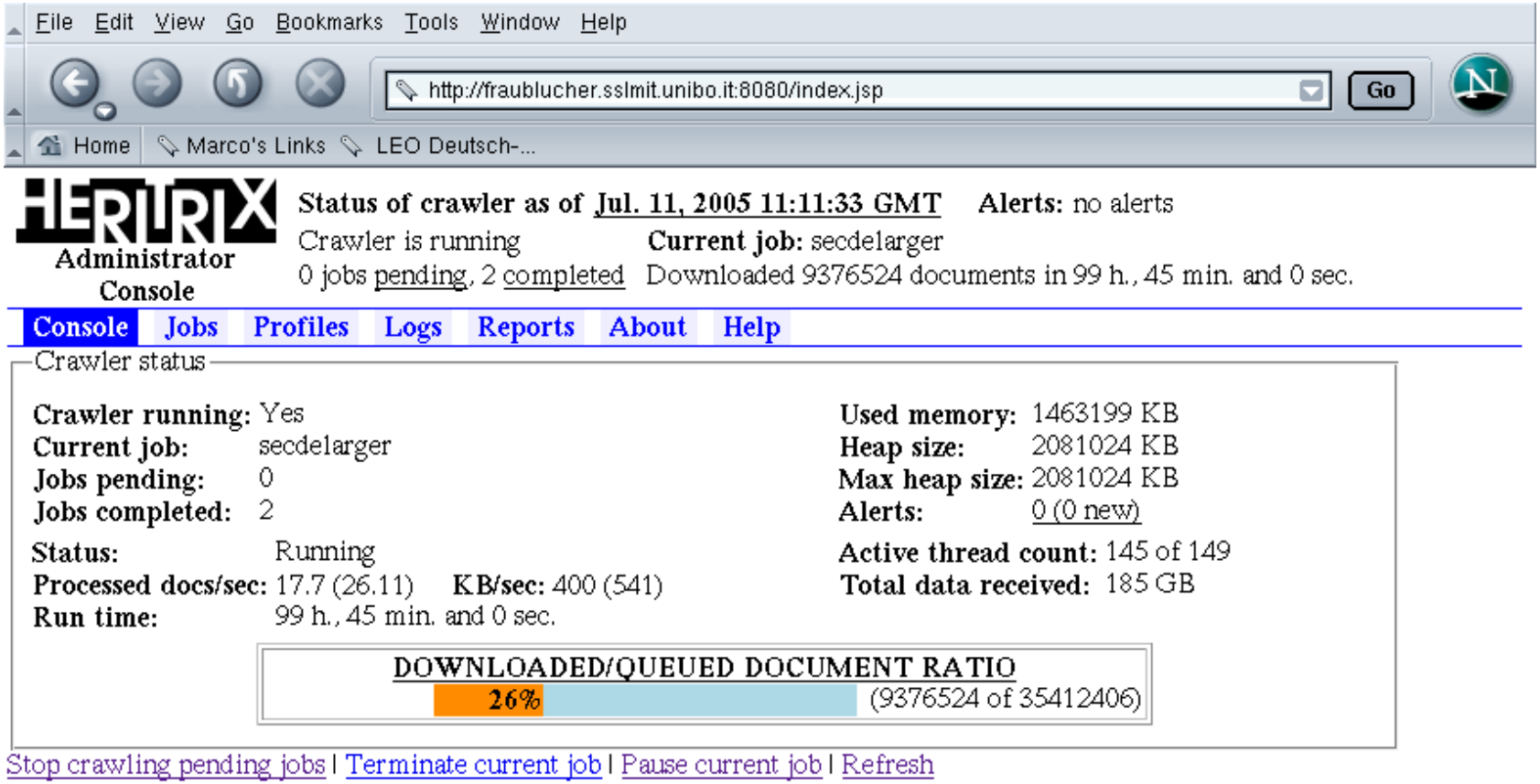
Heritrix

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

Heritrix

- `http://crawler.archive.org/`
- Free/open crawler of Internet Archive
- Very active, supporting community...
- that includes linguists and machine learning experts

The Heritrix WUI



The outpuf 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

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:

Language identification

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

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).

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

Near-duplicate spotting

- Simplified version of shingling algorithm of: Broder, Glassman, Manasse and Zweig (1997). Syntactic Clustering of the Web. Sixth International World-Wide Web Conference.
- 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

Annotation

- 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?

Conclusion

- 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?