An Architecture for Selective Web Harvesting: The Use Case of Heritrix

Vassilis Plachouras¹, Florent Carpentier², Julien Masanes², Thomas Risse³, Pierre Senellart⁴, Patrick Siehndel³ & Yannis Stavrakas¹

¹ IMIS, ATHENA Research Center, Athens, Greece
² Internet Memory Foundation, 93100 Montreuil, France
³ L3S Research Center, University of Hannover, Germany
⁴ Institut Mines-Télécom; Télécom ParisTech; CNRS LTCI, Paris, France
Motivation

• ARCOMEM Objectives
  • Selective preservation of Web content
  • Guided by social media

• Challenges for crawlers
  • Integration of computationally intensive content analysis
  • Accurate prioritization of links
  • Scalability
Large-scale distributed crawler

- Scalable crawling of Web pages and storing in HBase repository
  - Crawling at fast rate from start
  - Slowing down as little as possible after crawling billions of URLs
  - Observing politeness conventions
- Running multiple crawls concurrently
  - Each crawl job has own configuration and URL store
  - Single pool of fetchers for all crawl jobs
- Extensible configuration frameworks
  - Per-domain parameters
  - Updates are immediately usable

Developed by Internet Memory Foundation (IMF)
Application-Aware Helper

• Make the crawler aware of the particular kind of crawled Web applications
  • General classification of Web sites: Wiki, social network, blog, web forum, etc.
  • Technical implementation: Mediawiki, Wordpress, etc.
  • Specific instances: Twitter, CNN, etc.

• For each crawled document
  • Identify the corresponding Web application
  • If there’s a match, identifies the kind of Web page
  • Extract Web objects

• Handling template changes

Online Analysis

- Focus the crawler with respect to the crawl specification
- Integrate scores computed from different modules
  - Content written in a given language
  - URL matches regular expressions
  - URL corresponds to a known Web application identified by the AAH
  - URL contains keywords from crawl specification
  - ...
- Aggregating scores
  - Linear combination based on a weight vector
  - Automatically adjusting weights
Adapting Heritrix to ARCOMEM Crawling Architecture

• Typical crawling process
  • URL priority is set at scheduling time
  • Centralized crawler

• Integrating Heritrix
  • Adaptive prioritization
  • Scheduling links by remote processes

• Other modifications
  • writing content directly to RDBMS, HBase
  • Timed-closing of WARC files
  • Extracting anchor text with URLs
Adaptive prioritization

• Default frontier employs Berkeley DB hash table for storing URLs to crawl
  • Key for a URL in the hash table
    • Domain
    • Schedule Directive
    • Precedence
    • Counter
      | Domain | Schedule Directive | Precedence | Counter |
      |--------|--------------------|------------|---------|
      | 1 byte | 1 byte             | 6 bytes    |         |
  • Method next() returns only the next URL to crawl from a domain

• Our solution
  • Maintain a mapping from URL to key in hash table
  • Update priority by recalculating key
  • Counter ensures that there are no collisions
Scheduling links

- Need to allow external processes, possibly on different servers, to schedule links
- Heritrix Action directory
  - Need for additional scripts to write files on Heritrix local filesystem
- Our solution
  - Implemented a Web service that accepts JSON formatted links and priorities
Assessing the impact of decoupled URL fetching and prioritization

- Simulations to evaluate how adaptive and asynchronous/batch prioritization affects performance of a focused crawler

- Compare the effectiveness of a best-first crawler to
  - Adaptive prioritization
  - Asynchronous and batch URI prioritization
Baseline crawler

• For link \( s \rightarrow d \), the score of destination URL \( d \):
  
  • \((1-aw) \cdot \text{cosine}(\text{topic}, w_s)\)
  • \(aw \cdot \text{cosine}(\text{topic}, a_{s \rightarrow d})\) (1)

• where
  
  • \( w_s \) is the term vector of Web page \( s \)
  • \( a_{s \rightarrow d} \) is the term vector of the link’s anchor text
  • \( \text{topic} \) is the term vector of the crawl’s seeds
  • \( aw = 0.5 \) is the weight of anchor text

• Balance per queue
  
  • Crawl \( k \) pages from the site with the highest priority
Adaptive prioritization

- Update priority of already scheduled URIs when new links are found

- Score for URL $u_d$ is updated with one of the functions
  - MAX
  - AVG
  - SUM
  - LAST: keeps the most recent score
  - FIRST: equivalent to best-first crawler
Asynchronous & Batch Prioritization

- Fetch Web pages and write to the Web page repository
- Prioritize URLs in batches after crawling N pages
  - N = 100 pages
  - Link score computed from all anchors and source Web pages in which URL \( d \) is found
- Send batch of prioritized URLs back to the crawler
Simulation experiments

Simulation on 3 DMOZ topics
   Genetics, Recycling, Oceanography
Running simulated crawl
   Start from set of 20 randomly selected seeds (repeated 3 times)
   Topic vector is the sum of the seed vectors
   Crawl 10,000 web pages
Evaluation measures
   Harvest ratio: # of Web pages with cosine similarity to topic > 0.333
   Average similarity of crawled pages
   Fraction of DMOZ topics with at least one crawled page
Adaptive prioritization results

Using anchor text weight $aw = 0.5$

<table>
<thead>
<tr>
<th>Update function</th>
<th>Harvest Ratio</th>
<th>Average Similarity</th>
<th>DMOZ topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST</td>
<td>0.3317</td>
<td>0.2945</td>
<td>0.4979</td>
</tr>
<tr>
<td>AVG</td>
<td><strong>0.3609</strong></td>
<td>0.3024</td>
<td>0.5779</td>
</tr>
<tr>
<td>MAX</td>
<td>0.3388</td>
<td>0.2967</td>
<td>0.5270</td>
</tr>
<tr>
<td>SUM</td>
<td>0.2679</td>
<td>0.2759</td>
<td>0.4650</td>
</tr>
<tr>
<td>LAST</td>
<td>0.3404</td>
<td>0.2961</td>
<td><strong>0.5985</strong></td>
</tr>
</tbody>
</table>

- AVG and LAST have highest harvest ratios and find most pages from DMOZ topics
Simulation results: asynchronous & batch priorities

Parameters: \( aw = 0.5 \) and balance per queue \( bpq = 1 \) and 5

<table>
<thead>
<tr>
<th>Batch</th>
<th>Balance per queue</th>
<th>Update function</th>
<th>Harvest Ratio</th>
<th>Average Similarity</th>
<th>DMOZ topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>1</td>
<td>FIRST</td>
<td>0.3317</td>
<td>0.2945</td>
<td>0.4979</td>
</tr>
<tr>
<td>No</td>
<td>5</td>
<td>FIRST</td>
<td>0.2948</td>
<td>0.2819</td>
<td>0.4677</td>
</tr>
<tr>
<td>No</td>
<td>1</td>
<td>AVG</td>
<td><strong>0.3609</strong></td>
<td>0.3024</td>
<td>0.5779</td>
</tr>
<tr>
<td>No</td>
<td>5</td>
<td>AVG</td>
<td>0.3200</td>
<td>0.2897</td>
<td>0.5420</td>
</tr>
<tr>
<td>Yes</td>
<td>1</td>
<td>AVG</td>
<td>0.3556</td>
<td>0.3013</td>
<td>0.5260</td>
</tr>
<tr>
<td>Yes</td>
<td>5</td>
<td>AVG</td>
<td>0.3347</td>
<td>0.2952</td>
<td>0.5176</td>
</tr>
</tbody>
</table>

- Batch prioritization achieves similar harvest ratio to synchronous with adaptively averaging priorities (see rows 3 & 5)
- When increasing \( bpq \) from 1 to 5
  - synchronous crawler’s harvest ratio reduces ~ 11% (see rows 1 & 2 and 3 & 4)
  - Batch crawler’s harvest ratio reduces ~ 6%, more effective than synchronous (see rows 5 & 6 and 4 & 6)
Concluding remarks

• Effective architecture for selective crawling
  • Application-Aware Help
  • Online analysis
  • Large-scale distributed crawler
• Adapting Heritrix
  • Adaptive prioritization
  • Scheduling links from external processes
Thank you!

Vassilis Plachouras
Institute for the Management of Information Systems (IMIS)
ATHENA Research Center
Artemidos 6 & Epidavrou
Marousi 15125, Athens
Greece

Email: vplachouras@imis.athena-innovation.gr
Phone: +30 210 6875 413