CES Data Scientist

Web Crawling

19 October 2015

Plan

Discovering new URLs

Identifying duplicates

Crawling architecture

Recrawling URLs?

Modern Crawling

Conclusion

Web Crawlers

- crawlers, (Web) spiders, (Web) robots: autonomous user agents that retrieve pages from the Web
- Basics of crawling:
  1. Start from a given URL or set of URLs
  2. Retrieve and process the corresponding page
  3. Discover new URLs (cf. next slide)
  4. Repeat on each found URL
- No real termination condition (virtual unlimited number of Web pages!)
- Graph-browsing problem

  deep-first: not very adapted, possibility of being lost in robot traps

  breadth-first

  combination of both: breadth-first with limited-depth deep-first on each discovered website

Sources of new URLs

- From HTML pages:
  - hyperlinks `<a href="...">...</a>`
  - media `<img src="...">` `<embed src="...">` `<object data="...">`
  - frames `<frame src="...">` `<iframe src="...">`
  - JavaScript links `window.open("...")`
  - etc.
- Other hyperlinked content (e.g., PDF files)
- Non-hyperlinked URLs that appear anywhere on the Web (in HTML text, text files, etc.): use regular expressions to extract them
- Referrer URLs
- Sitemaps [sitemaps.org, 2008]

Scope of a crawler

- Web-scale
  - The Web is infinite! Avoid robot traps by putting depth or page number limits on each Web server
  - Focus on important pages [Abiteboul et al., 2003]
- Web servers under a list of DNS domains: easy filtering of URLs
- A given topic: focused crawling techniques [Chakrabarti et al., 1999, Diligenti et al., 2000] based on classifiers of Web page content and predictors of the interest of a link.
- The national Web (cf. public deposit, national libraries): what is this? [Abiteboul et al., 2002]
- A given Web site: what is a Web site? [Senellart, 2005]

Plan

TELECOM ParisTech

A word about hashing

Definition

A hash function is a deterministic mathematical function transforming objects (numbers, character strings, binary...) into fixed-size, seemingly random, numbers. The more random the transformation is, the better.

Example

Java hash function for the `String` class:

$$\sum_{i=0}^{n-1} s_i \times 31^{n-i-1} \bmod 2^{32}$$

where s_i is the (Unicode) code of character i of a string s.

Identification of duplicate Web pages

Problem

Identifying duplicates or near-duplicates on the Web to prevent multiple indexing

- **trivial duplicates:** same resource at the same canonized URL:

  ```
  http://example.com:80/toto
  http://example.com/titi/../toto
  ```

- **exact duplicates:** identification by hashing
- **near-duplicates:** (timestamps, tip of the day, etc.) more complex!

Near-duplicate detection

Edit distance. Count the minimum number of basic modifications (additions or deletions of characters or words, etc.) to obtain a document from another one. Good measure of similarity, and can be computed in $O(mn)$ where m and n are the size of the documents. But: does not scale to a large collection of documents (unreasonable to compute the edit distance for every pair!).

Shingles. Idea: two documents similar if they mostly share the same succession of k-grams (succession of tokens of length k).

Example

I like to watch the sun set with my friend.
My friend and I like to watch the sun set.

$S =${i like, like to, my friend, set with, sun set, the sun, to watch, watch the, with my}
$T =${and i, friend and, i like, like to, my friend, sun set, the sun, to watch, watch the}

Hashing shingles to detect duplicates [Broder et al., 1997]

- Similarity: Jaccard coefficient on the set of shingles:

$$J(S,T) = \frac{|S \cap T|}{|S \cup T|}$$

- Still costly to compute! But can be approximated as follows:
  1. Choose N different hash functions
  2. For each hash function h_i and each set of shingles $S_k = \{s_{k1} \dots s_{kn}\}$, store $\phi_{ik} = \min_j h_i(s_{kj})$
  3. Approximate $J(S_k, S_l)$ as the proportion of ϕ_{ik} and ϕ_{il} that are equal
- Possibly to repeat in a hierarchical way with super-shingles (we are only interested in very similar documents)

Plan

Crawling ethics

- Standard for robot exclusion: robots.txt at the root of a Web server [Koster, 1994].

```
User-agent: *
Allow: /searchhistory/
Disallow: /search
```

- Per-page exclusion.

```
<meta name="ROBOTS" content="NOINDEX,NOFOLLOW">
```

- Per-link exclusion.

```
<a href="toto.html" rel="nofollow">Toto</a>
```

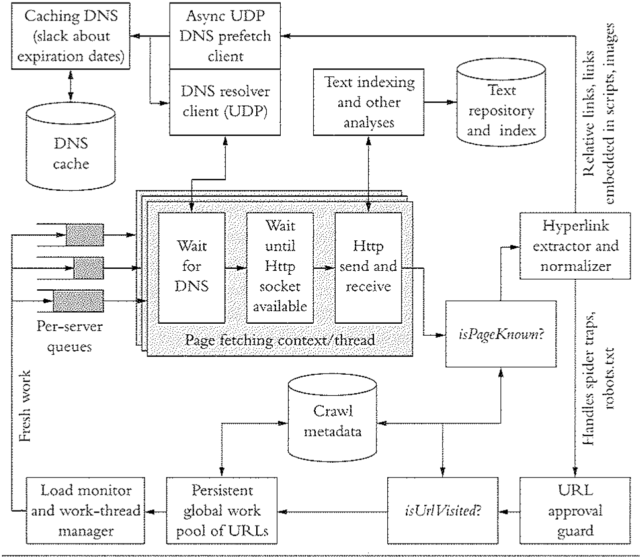
- Avoid Denial Of Service (DOS), wait ≈1s between two repeated requests to the same Web server

Parallel processing

Network delays, waits between requests:

- Per-server queue of URLs
- Parallel processing of requests to different hosts:
  - multi-threaded programming
  - asynchronous inputs and outputs (`select`, classes from `java.util.concurrent`): less overhead
- Use of keep-alive to reduce connexion overheads

General Architecture [Chakrabarti, 2003]



Plan

Discovering new URLs

Identifying duplicates

Crawling architecture

Recrawling URLs?

Modern Crawling

Conclusion

Refreshing URLs

- Content on the Web changes
- Different change rates:

  online newspaper main page: every hour or so
  published article: virtually no change
- Continuous crawling, and identification of change rates for adaptive crawling: how to know the time of last modification of a Web page?

Estimating the Freshness of a Page

1. Check HTTP timestamp.
2. Check content timestamp.
3. Compare a hash of the page with a stored hash.
4. Non-significant differences (ads, fortunes, request timestamp):
   - only hash text content, or “useful” text content;
   - compare distribution of *n*-grams (shingling);
   - or even compute edit distance with previous version.

Adapting strategy to each different archived website?

Plan

Discovering new URLs

Identifying duplicates

Crawling architecture

Recrawling URLs?

Modern Crawling
- Modern Web Sites
- Social Networking Sites

Conclusion

Plan

Crawling Modern Web Sites

- Some modern Web sites only work when cookies are activated (session cookies), or when JavaScript code is interpreted
- Regular Web crawlers (wget, Heritrix, Apache Nutch) do not usually perform any cookie management and do not interpret JavaScript code
- Crawling of some Websites therefore require more advanced tools

Advanced crawling tools

Web scraping frameworks such as scrapy (Python) or WWW::Mechanize (Perl) simulate a Web browser interaction and cookie management (but no JS interpretation)

Headless browsers such as htmlunit simulate a Web browser, including simple JavaScript processing

Browser instrumentors such as Selenium allow full instrumentation of a regular Web browser (Chrome, Firefox, Internet Explorer)

OXPath: a full-fledged navigation and extraction language for complex Web sites [Sellers et al., 2011]

Plan

Most popular Web sites

1	google.com
2	facebook.com
3	youtube.com
4	yahoo.com
5	baidu.com
6	wikipedia.org
7	live.com
8	twitter.com
9	qq.com
10	amazon.com
11	blogspot.com
12	linkedin.com
13	google.co.in
14	taobao.com
15	sina.com.cn
16	yahoo.co.jp
17	msn.com
18	wordpress.com
19	google.com.hk
20	t.co
21	google.de
22	ebay.com
23	google.co.jp
24	googleusercontent.com
25	google.co.uk
26	yandex.ru
27	163.com
28	weibo.com

(Alexa)

Most popular Web sites

1 google.com
2 facebook.com
3 youtube.com
4 yahoo.com
5 baidu.com
6 wikipedia.org
7 live.com
8 twitter.com
9 qq.com
10 amazon.com
11 blogspot.com
12 linkedin.com
13 google.co.in
14 taobao.com
15 sina.com.cn
16 yahoo.co.jp
17 msn.com
18 wordpress.com
19 google.com.hk
20 t.co
21 google.de
22 ebay.com
23 google.co.jp
24 googleusercontent.com
25 google.co.uk
26 yandex.ru
27 163.com
28 weibo.com

(Alexa)

Social networking sites

Most popular Web sites

1 google.com
2 facebook.com
3 youtube.com
4 yahoo.com
5 baidu.com
6 wikipedia.org
7 live.com
8 twitter.com
9 qq.com
10 amazon.com
11 blogspot.com
12 linkedin.com
13 google.co.in
14 taobao.com
15 sina.com.cn
16 yahoo.co.jp
17 msn.com
18 wordpress.com
19 google.com.hk
20 t.co
21 google.de
22 ebay.com
23 google.co.jp
24 googleusercontent.com
25 google.co.uk
26 yandex.ru
27 163.com
28 weibo.com

(Alexa)

Social networking sites

Sites with social networking features (friends, user-shared content, user profiles, etc.)

Social data on the Web

Huge numbers of users (2012):

Facebook	900 million
QQ	540 million
W. Live	330 million
Weibo	310 million
Google+	170 million
Twitter	140 million
LinkedIn	100 million

Social data on the Web

Huge numbers of users (2012):

Facebook	900 million
QQ	540 million
W. Live	330 million
Weibo	310 million
Google+	170 million
Twitter	140 million
LinkedIn	100 million

Huge volume of shared data:

250 million tweets per day on Twitter (3,000 per second on average!)...

... including statements by heads of states, revelations of political activists, etc.

Crawling Social Networks

- Theoretically possible to crawl social networking sites using a regular Web crawler
- Sometimes not possible:
  `https://www.facebook.com/robots.txt`
- Often very inefficient, considering politeness constraints
- Better solution: Use provided social networking APIs
  `https://dev.twitter.com/docs/api/1.1`
  `https://developers.facebook.com/docs/graph-api/reference/v2.1/`
  `https://developer.linkedin.com/apis`
  `https://developers.google.com/youtube/v3/`
- Also possible to buy access to the data, directly from the social network or from brokers such as `http://gnip.com/`

Social Networking APIs

- Most social networking Web sites (and some other kinds of Web sites) provide APIs to effectively access their content
- Usually a RESTful API, occasionally SOAP-baed
- Usually require a token identifying the application using the API, sometimes a cryptographic signature as well
- May access the API as an authenticated user of the social network, or as an external party
- APIs seriously limit the rate of requests: `https://dev.twitter.com/docs/api/1.1/get/search/tweets`

REST

- Mode of interaction with a Web service
- Follow the KISS (Keep it Simple, Stupid) principle
- Each request to the service is a simple HTTP GET method
- Base URL is the URL of the service
- Parameters of the service are sent as HTTP parameters (in the URL)
- HTTP response code indicates success or failure
- Response contains structured output, usually as JSON or XML
- No side effect, each request independent of previous ones

The Case of Twitter

- Two main APIs:
  - REST APIs, including search, getting information about a user, a list, followers, etc. `https://dev.twitter.com/docs/api/1.1`
  - Streaming API, providing real-time result
- Very limited history available
- Search can be on keywords, language, geolocation (for a small portion of tweets)

Cross-Network Crawling

- Often useful to combine results from different social networks
- Numerous libraries facilitating SN API accesses (twipy, Facebook4J, FourSquare VP C++ API...) incompatible with each other... Some efforts at generic APIs (OneAll, APIBlender [Gouriten and Senellart, 2012])
- Example use case: No API to get all check-ins from FourSquare, but a number of check-ins are available on Twitter; given results of Twitter Search/Streaming, use FourSquare API to get information about check-in locations.

Plan

Conclusion

What you should remember

- Crawling as a graph-browsing problem.
- Shingling for identifying duplicates.
- Numerous engineering issues in building a Web-scale crawler.
- Crawling modern Web content is not as easy as launching a traditional Web crawler
- Often critical to focus the crawl towards content of interest
- Ideally: a traditional large-scale crawler that knows when to delegate to more specialized crawling mechanisms (tools querying social networking APIs, deep Web crawlers, JS-aware crawlers, etc.)
- Huge variety of tools, techniques, suitable for different needs

References

Software

- Wget, a simple yet effective Web spider (free software)
- Heritrix, a Web-scale highly configurable Web crawler, used by the Internet Archive (free software)
- HTML Parser, TagSoup: Java libraries for parsing real-world Web pages

To go further

- A good textbook [Chakrabarti, 2003]
- Main references:
  - HTML 4.01 recommendation [W3C, 1999]
  - HTTP/1.1 RFC [IETF, 1999]

Bibliography I

Serge Abiteboul, Grégory Cobena, Julien Masanès, and Gerald Sedrati. A first experience in archiving the French Web. In *Proc. ECDL*, Roma, Italie, September 2002.

Serge Abiteboul, Mihai Preda, and Gregory Cobena. Adaptive on-line page importance computation. In *Proc. WWW*, May 2003.

Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the Web. *Computer Networks*, 29(8-13):1157–1166, 1997.

Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, San Fransisco, USA, 2003.

Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: A new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11–16):1623–1640, 1999.

Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused crawling using context graphs. In *Proc. VLDB*, Cairo, Egypt, September 2000.

Bibliography II

Georges Gouriten and Pierre Senellart. API Blender: A uniform interface to social platform APIs. In *Proc. WWW*, Lyon, France, April 2012. Developer track.

IETF. Request For Comments 2616. Hypertext transfer protocol—HTTP/1.1. `http://www.ietf.org/rfc/rfc2616.txt`, June 1999.

Martijn Koster. A standard for robot exclusion. `http://www.robotstxt.org/orig.html`, June 1994.

Andrew Sellers, Tim Furche, Georg Gottlob, Giovanni Grasso, and Christian Schallhart. Exploring the Web with OXPath. In *LWDM*, 2011.

Pierre Senellart. Identifying Websites with flow simulation. In *Proc. ICWE*, pages 124–129, Sydney, Australia, July 2005.

sitemaps.org. Sitemaps XML format. `http://www.sitemaps.org/protocol.php`, February 2008.

Bibliography III

W3C. HTML 4.01 specification, September 1999.
http://www.w3.org/TR/REC-html40/.

Licence de droits d'usage

Contexte public } avec modifications