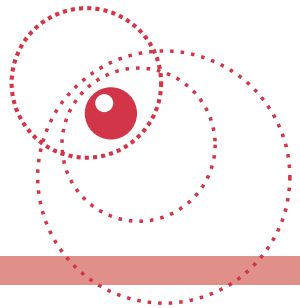


INF344: Données du Web

Web Crawling

3 mai 2016

TELECOM ParisTech

Plan

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

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

Legal aspects (France) – 1/2

- General principles:
  - to access or keep access to a “system for automated data processing” *in a fraudulous manner* is punished of two years of prison and 60,000 euros fine (Code pénal 323-1, modified by law 2015-912 on “Renseignement”)
  - to disrupt the functioning of a “system for automated data processing” is punished of five years of prison and 150,000 euros fine, extended to seven years and 300,000 euros when the system is a public one containing personal information (Code pénal 323-2, modified by law 2015-912 on “Renseignement”)
- A Web site hosted in a different country may invoke completely different legal principles, under a different jurisdiction
- Crawling content can be considered accessing and keeping access to a “system for automated data processing” (Cour d’appel de Paris, 5 February 2014, “Bluetouff case”)

Legal aspects (France) – 2/2

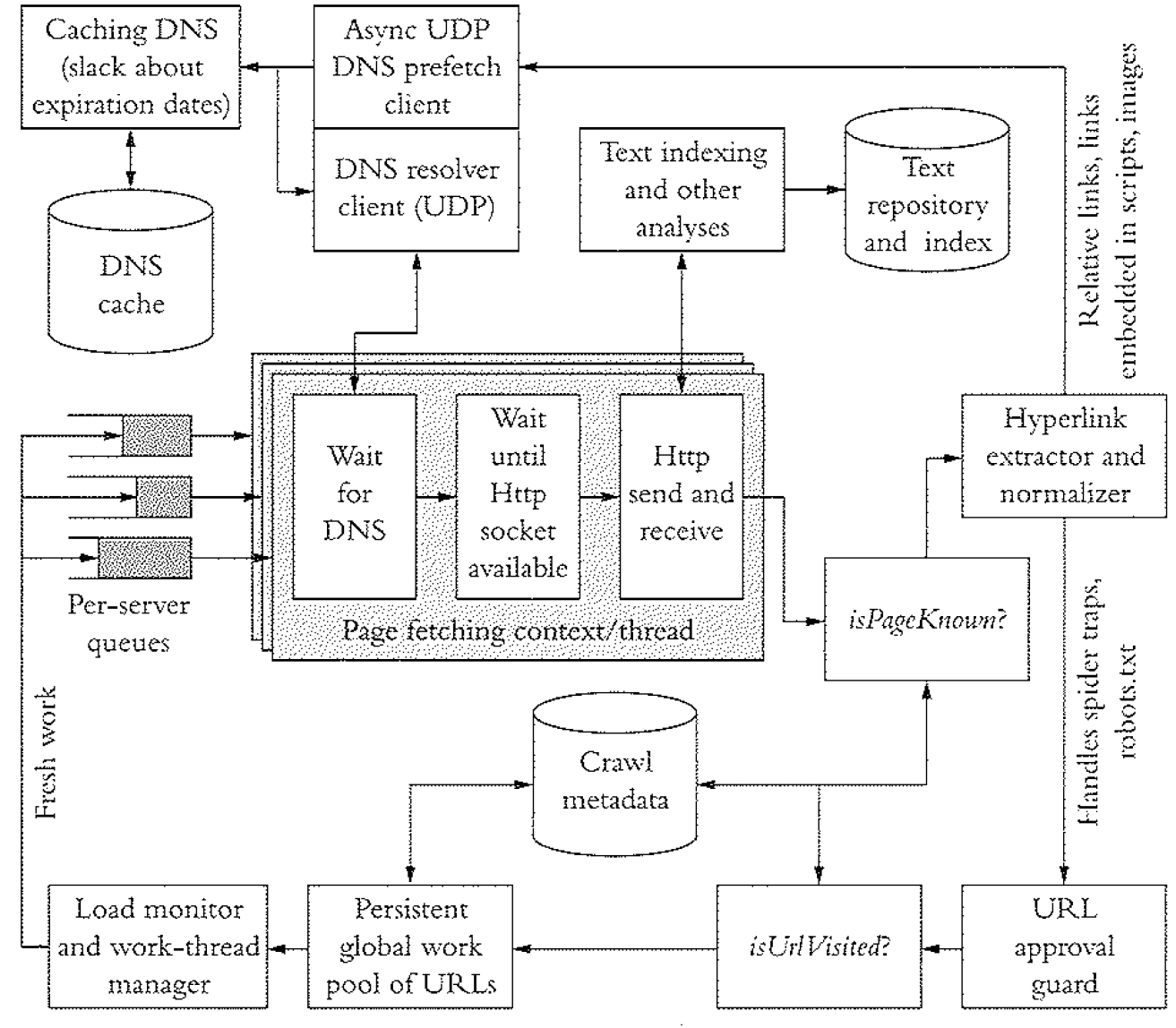
- robots.txt files are a de facto standard, and instructions in robots.txt files can be taken are a receivable way to specify what can be crawled (Cour d'appel de Paris, 26 January 2011, Google vs SAIF)
- Frequent requests to a Web site can be considered as a way to disrupt the functioning of a "system for automated data processing" (Cour d'appel de Bordeaux, 15 November 2011, Cédric M. vs C-Discount), but only if it reaches abusive levels and can be shown to have cause disruption
- Web content is subject to "droit d'auteur" (Code de la propriété intellectuelle, Première partie, Livre Ier) and cannot generally be broadcast by third-parties; only transient copies are allowed (CJEU, 5 June 2014, PRCA vs NLA)
- Web content containing personal data is even more sensitive (Loi "Informatique et Libertés"): personal data should be collected for a specific purpose, and should be kept updated

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]



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

Plan

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

Templated Web Site

- Many Web sites (especially, Web forums, blogs) use one of a few content management systems (CMS)
- Web sites that use the same CMS will be similarly structured, present a similar layout, etc.
- Information is somewhat structured in CMSs: publication date, author, tags, forums, threads, etc.
- Some structure differences may exist when Web sites use different versions, or different themes, of a CMS

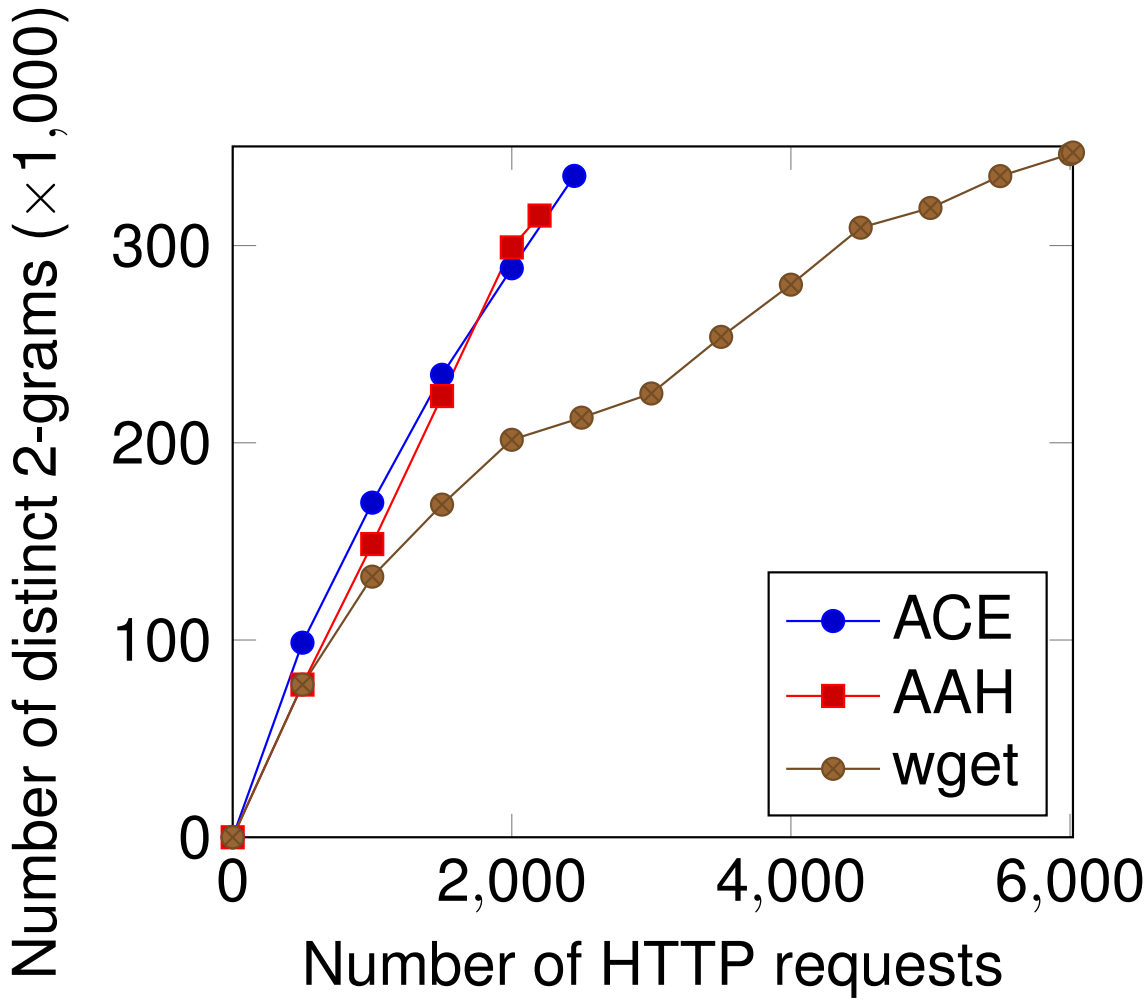
Crawling CMS-Based Web Sites

- Traditional crawling approaches crawl Web sites independently of the nature of the sites and of their CMS
- When the CMS is known:
  - Potential for much more efficient crawling strategies (avoid pages with redundant information, uninformative pages, etc.)
  - Potential for automatic extraction of structured content
- Two ways of approaching the problem:
  - Have a handcrafted knowledge base of known CMSs, their characteristics, how to crawl and extract information [Faheem and Senellart, 2013b,a] (AAH)
  - Automatically infer the best way to crawl a given CMS [Faheem and Senellart, 2014] (ACE)
- Need to be robust w.r.t. template change

Detecting CMSs

- One main challenge in intelligent crawling and content extraction is to identify the CMS and then perform the best crawling strategy accordingly
- Detecting CMS using:
  1. URL patterns,
  2. HTTP metadata,
  3. textual content,
  4. XPath patterns, etc.
- These can be manually described (AAH), or automatically inferred (ACE)
- For instance the vBulletin Web forum content management system, that can be identified by searching for a reference to a `vbulletin_global.js` JavaScript script by using a simple `//script/@src` XPath expression.

Crawling http://www.rockamring-blog.de/ [Faheem and Senellart, 2014]



Plan

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

The Deep Web

Definition (Deep Web, Hidden Web, Invisible Web)

All the content on the Web that is not directly accessible through hyperlinks. In particular: HTML forms, Web services.

Size estimate: 500 times more content than on the surface Web! [BrightPlanet, 2000]. Hundreds of thousands of deep Web databases [Chang et al., 2004]

Sources of the Deep Web

Example

- *Yellow Pages* and other directories;
- Library catalogs;
- Weather services;
- US Census Bureau data;
- etc.

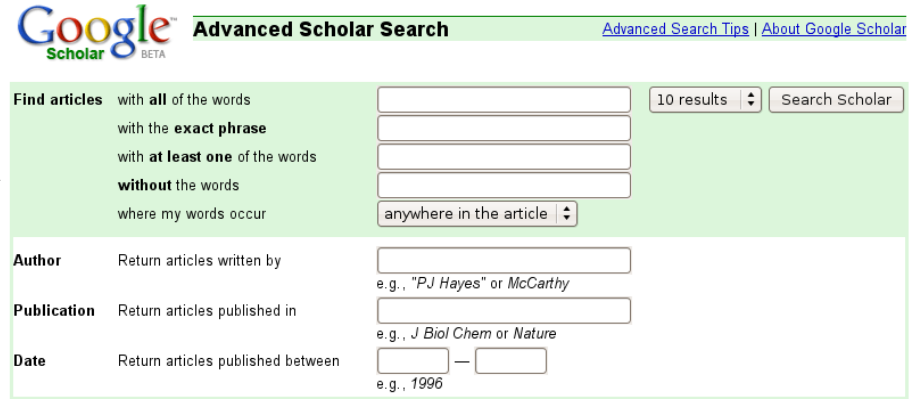
Discovering Knowledge from the Deep Web [Nayak et al., 2012]

- Content of the deep Web hidden to classical Web search engines (they just follow links)
- But very valuable and high quality!
- Even services allowing access through the surface Web (e.g., e-commerce) have more semantics when accessed from the deep Web
- How to benefit from this information?
- How to analyze, extract and model this information?

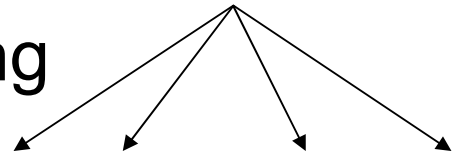
Focus here: Automatic, unsupervised, methods, for a given domain of interest

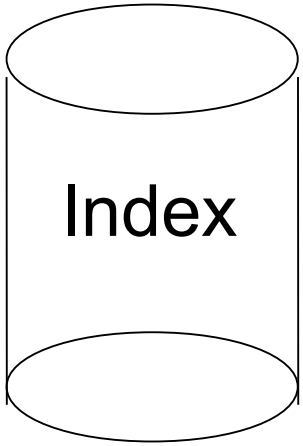
Extensional Approach

discovery

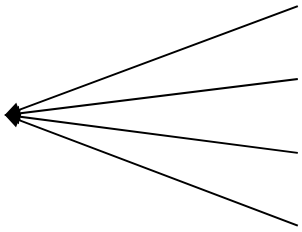


siphoning





indexing



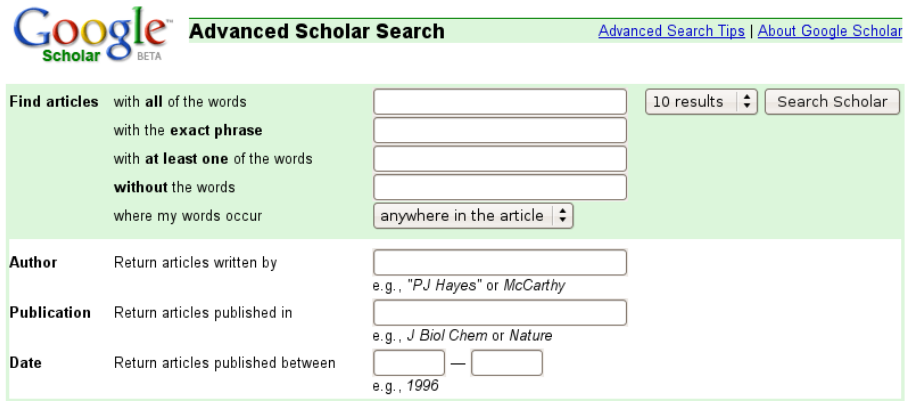
bootstrap

Notes on the Extensional Approach

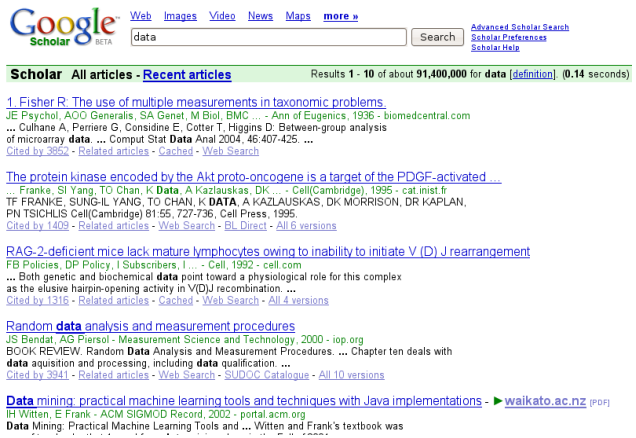
- Main issues:
  - Discovering services
  - Choosing appropriate data to submit forms
  - Use of data found in result pages to bootstrap the siphoning process
  - Ensure good coverage of the database
- Approach favored by Google, used in production [Madhavan et al., 2006]
- Not always feasible (huge load on Web servers)

Intensional Approach

discovery



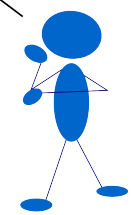
probing



analyzing

Form wrapped as a Web service

query



Notes on the Intensional Approach

- More ambitious [Chang et al., 2005, Senellart et al., 2008]
- Main issues:
  - Discovering services
  - Understanding the structure and semantics of a form
  - Understanding the structure and semantics of result pages
  - Semantic analysis of the service as a whole
  - Query rewriting using the services
- No significant load imposed on Web servers

A Quirky Deep Web

- Numerous works on form understanding and information extraction from the deep Web [He et al., 2007, Varde et al., 2009, Khare et al., 2010]
- Formal models for answering queries under access pattern restrictions [Li and Chang, 2001, Calì and Martinenghi, 2008, Calì and Martinenghi, 2010, Benedikt et al., 2012]
- Siphoning of hidden Web databases [Barbosa and Freire, 2004, Jin et al., 2011, Sheng et al., 2012]

- Those works ignore lots of quirky dimensions of deep Web interfaces
- Here: towards a more comprehensive framework for deep Web modeling and querying

Views

Deep Web sources offer views over (most often relational) data, through, at the very least:

- selection (depending on user's query, or implicit in the service), in particular inequalities
- projection (not available attributes are exported by a given service)

And also (but less critically):

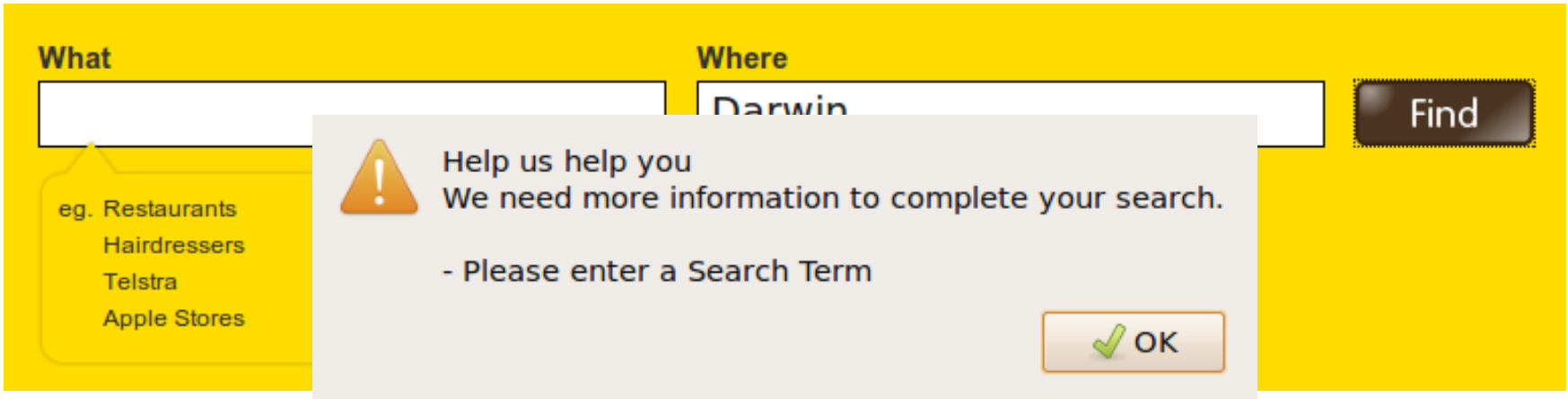
- joins (quite common in a Web application – but from an outsider's perspective, often enough to see the result of a join as the relation of interest)
- union, intersection, difference, etc. (relatively rare)
- aggregation (usually not the most important part of the service)
- more complex processing (rare in practice)

Limited access patterns

Australian Yellow Pages search form:

Limited access patterns

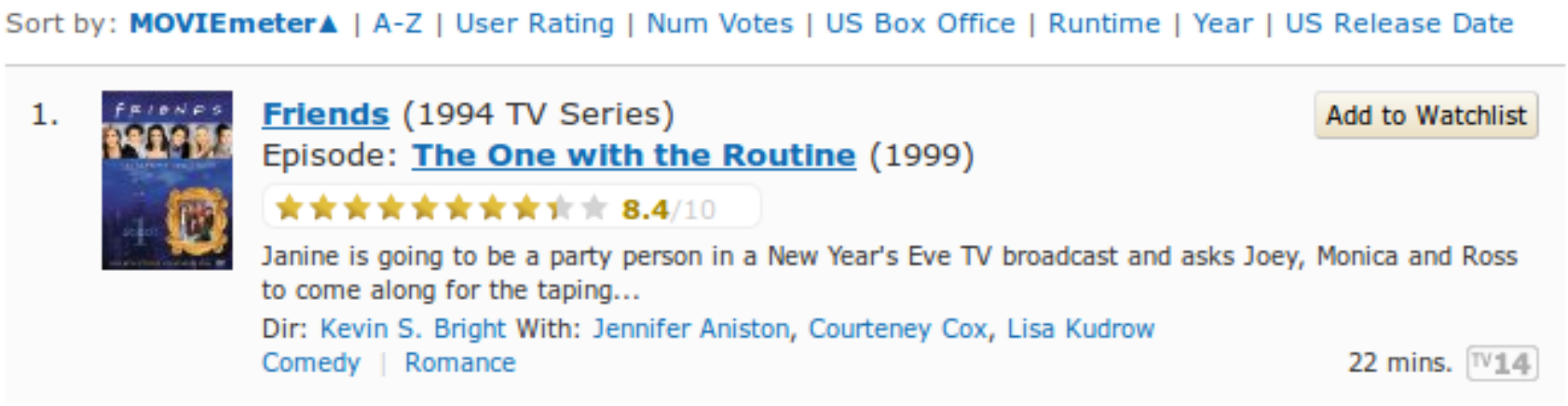
Australian Yellow Pages search form:



Required attributes, dependencies between attributes of the form, etc.

Ranking of results

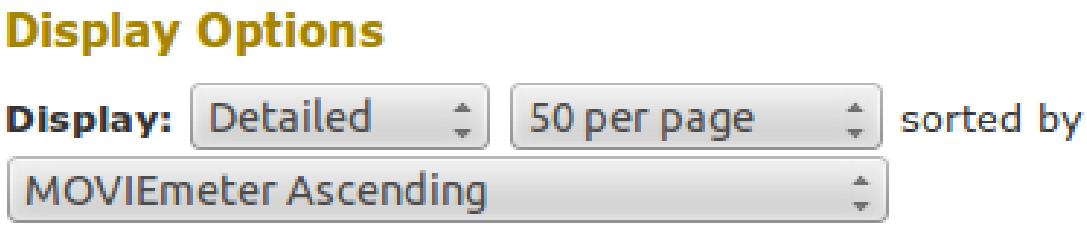
IMDb advanced search sort criteria:



Different possible sort criteria, some according to non-exported attributes

Paging

Paging in IMDb:



10,001-10,050 of 100,289 titles. « Prev Next »

Each page of results requires a separate network access, and therefore has a cost

Overflow

What you get when you try to access the 100,001-th result to an IMDb advanced query:

Error

Sorry, IMDb does not serve more than 100000 results for any query. (You asked for results starting from 100001)

Only a (top-ranked) subset of the results is available for each access

Policy limitations

Twitter API rate limitation:

REST API Rate Limiting

The default rate limit for calls to the REST API varies depending on the authorization method being used and whether the method itself requires authentication.

- Unauthenticated calls are permitted 150 requests per hour. Unauthenticated calls are measured against the public facing IP of the server or device making the request.
- OAuth calls are permitted 350 requests per hour and are measured against the `oauth_token` used in the request.

Limited rate of queries per minute, hour, query... Several services of the same source may share the same limits.

Incomplete information: Projection

Several views of the same information on IMDB:

Incomplete information: Projection

Several views of the same information on IMDB:

1. It's a Wonderful Life (1946)
   - aka "Frank Capra's It's a Wonderful Life" - USA *(complete title)*
   - aka "La vie est belle" - Belgium *(French title)*, Canada *(French title)*, France
   - aka "¡Qué bello es vivir!" - Peru *(imdb display title)*, Spain
   - aka "Ist das Leben nicht schön?" - Austria *(TV title)*, West Germany *(TV title)*
   - aka "¡Que bello es vivir!" - Uruguay
   - aka "A Felicidade Não Se Compra" - Brazil
   - aka "Az élet csodaszép" - Hungary
   - aka "Det er herligt at leve" - Denmark
   - aka "Divan život" - Serbia
   - aka "Divan zivot" - Yugoslavia *(Croatian title)* *(imdb display title)*
   - aka "Do Céu Caiu Uma Estrela" - Portugal
   - aka "Ihmeellinen on elämä" - Finland
   - aka "La vita è meravigliosa" - Italy
   - aka "Livet är underbart" - Sweden
   - aka "Livet er vidunderlig" - Norway *(imdb display title)*
   - aka "Mens, durf te leven" - Netherlands *(informal literal title)*
   - aka "Mia yperohi zoi" - Greece *(transliterated ISO-LATIN-1 title)*
   - aka "O viata minunata" - Romania *(imdb display title)*
   - aka "Qué bello es vivir" - Argentina
   - aka "Que bonic és viure!" - Spain *(Catalan title)*
   - aka "Que la vie est belle" - Belgium *(French title)*
   - aka "Sahane hayat" - Turkey *(Turkish title)* *(DVD title)*
   - aka "Subarashiki kana, jinsei!" - Japan
   - aka "To wspaniale zycie" - Poland
   - aka "Wat een mooi leven" - Belgium *(Flemish title)*
   - aka "Zycie jest cudowne" - Poland

Incomplete information: Projection

Several views of the same information on IMDB:

1.  **It's a Wonderful Life** (1946) — Add to Watchlist
★★★★★★★★★½ 8.7/10
An angel helps a compassionate but despairingly frustrated businessman by showing what life would have been like if he never existed.
Dir: Frank Capra With: James Stewart, Donna Reed, Lionel Barrymore
Drama | Fantasy — 130 mins. UR

2.  **It Happened One Night** (1934) — Add to Watchlist
★★★★★★★★½☆ 8.3/10
A spoiled heiress, running away from her family, is helped by a man who's actually a reporter looking for a story.
Dir: Frank Capra With: Clark Gable, Claudette Colbert, Walter Connolly
Comedy | Romance — 105 mins. UR

3.  **Mr. Smith Goes to Washington** (1939) — Add to Watchlist
★★★★★★★★½☆ 8.4/10
A naive man is appointed to fill a vacancy in the US Senate. His plans promptly collide with political corruption, but he doesn't back down.
Dir: Frank Capra With: James Stewart, Jean Arthur, Claude Rains
Comedy | Drama — 129 mins. Approved

Same relation(s), different attributes projected out

Incomplete information: Granularity

Release date API on IMDb:

Release dates for

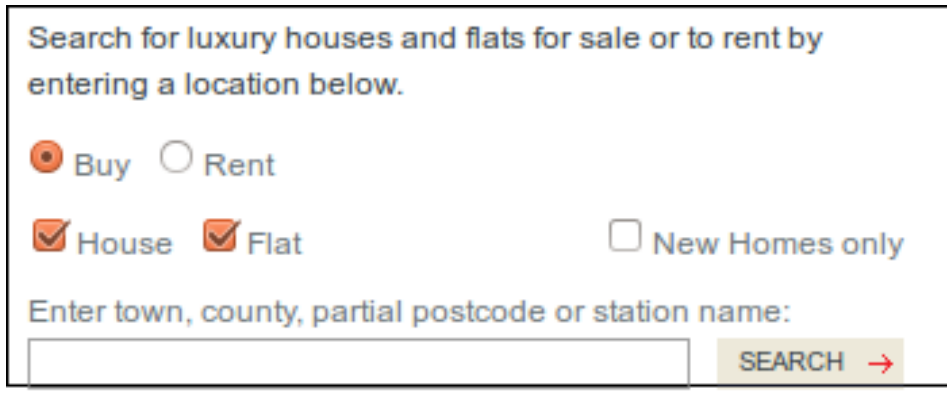
It's a Wonderful Life (1946) More at IMDbPro »

Country	Date
USA	20 December 1946 (New York City, New York)

The granularity of the presented information may not be the most precise one

Recency

Savills property search:

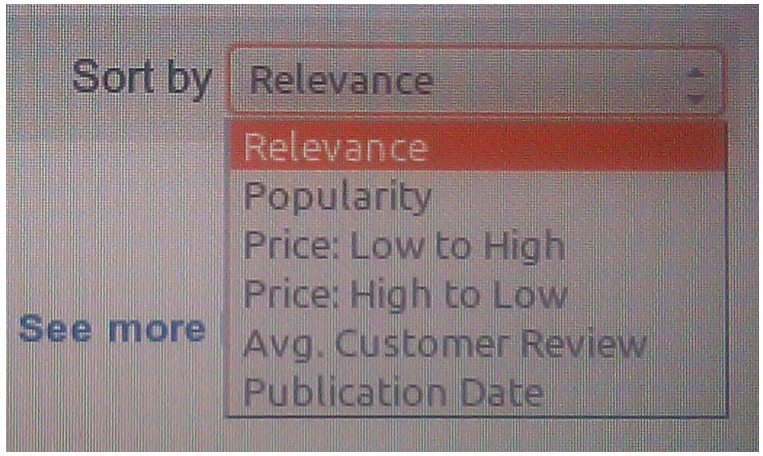


Publication time is a special attribute of interest:

- may or may not be exported
- may or may not be queriable (sometimes in a very weird way!)
- often used as a ranking criterion
- granularity plays an important role
- publication date $<$ query date

Uncertainty in the ranking

Amazon Books sorting options:



- Proprietary ranking functions
- Weighted combination of attributes with unknown weights [Soliman et al., 2011]
- Ranking according to an unexported attribute

Dependencies across services

Some of IMDb advanced search options:

Advanced Title Search

Want to get a list of comedies from the 1970s that have at least 1000 votes and an average rating of 7.5 or higher? Use Advanced Title Search.

Advanced Name Search

Want a list of males in the database who are Virgos and over 6 feet tall? Use Advanced Name Search.

Collaborations and Overlaps

Want a list of titles in which both Brad Pitt and George Clooney appeared? Or a list of people who worked on both Forrest Gump and Apollo 13? Try searching Collaborations and Overlaps.

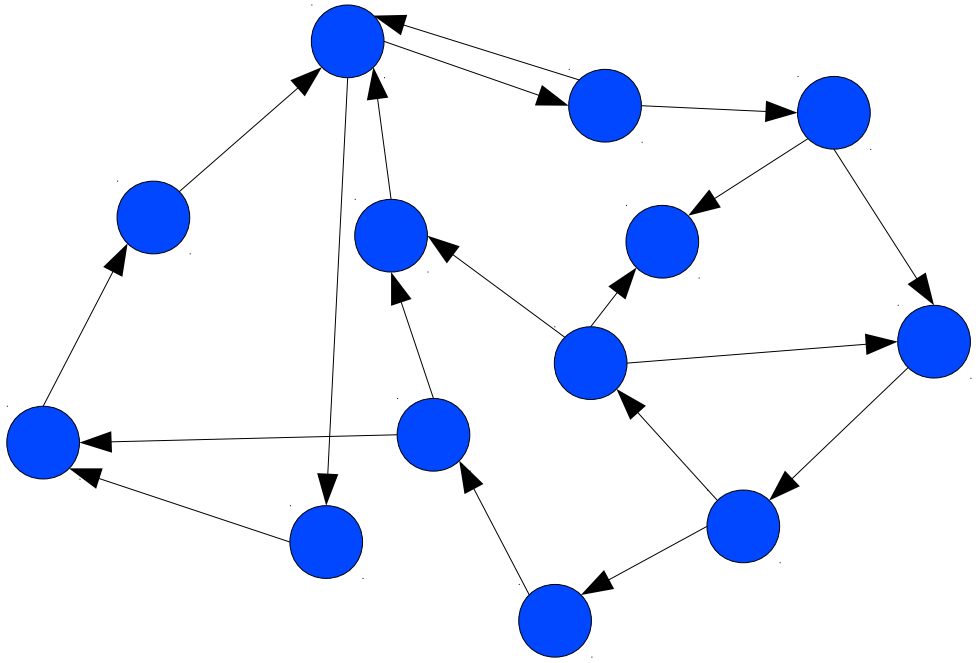
- services of the same source provide different correlated views of the same data
- dependencies (inclusion) across services are common too
- a given service often satisfies some key dependencies

But also…

- non-conjunctive forms (common in digital library applications)
- unknown characteristics of information retrieval systems (keyword querying vs exact querying, indexing of stop words, stemming used, etc.)
- intricate interactions (AJAX autocompletion, submitting a form as a first step before submitting another form, etc.)
- potential side effects of a service

Plan

A directed graph



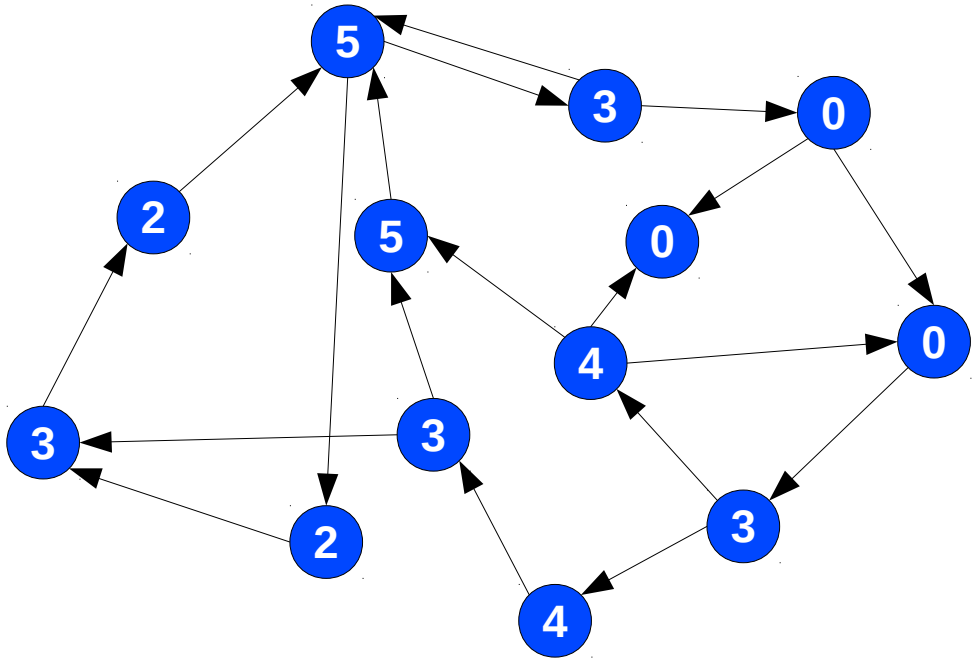
Web

Social network

P2P

etc.

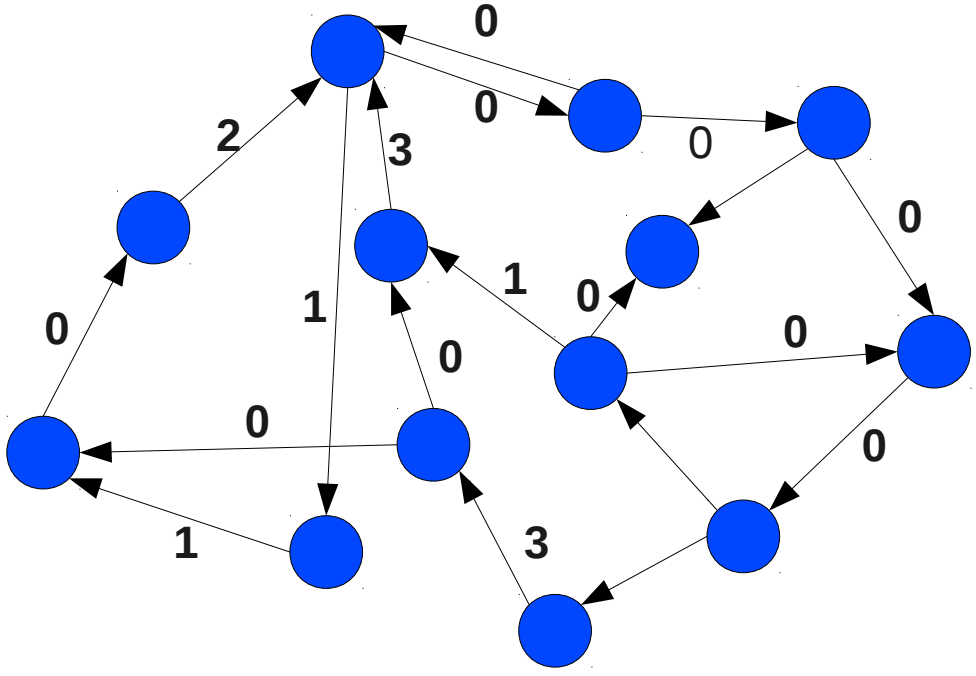
Weighted



Let *u* be a node,

β(u) = count of the word *Bhutan* in all the tweets of u

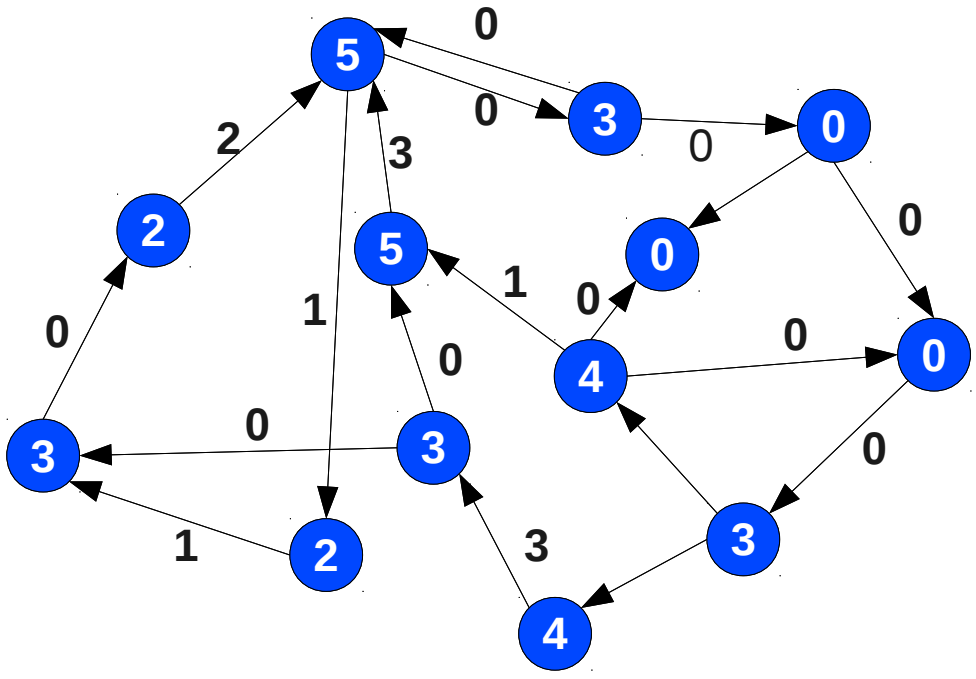
Even more weighted



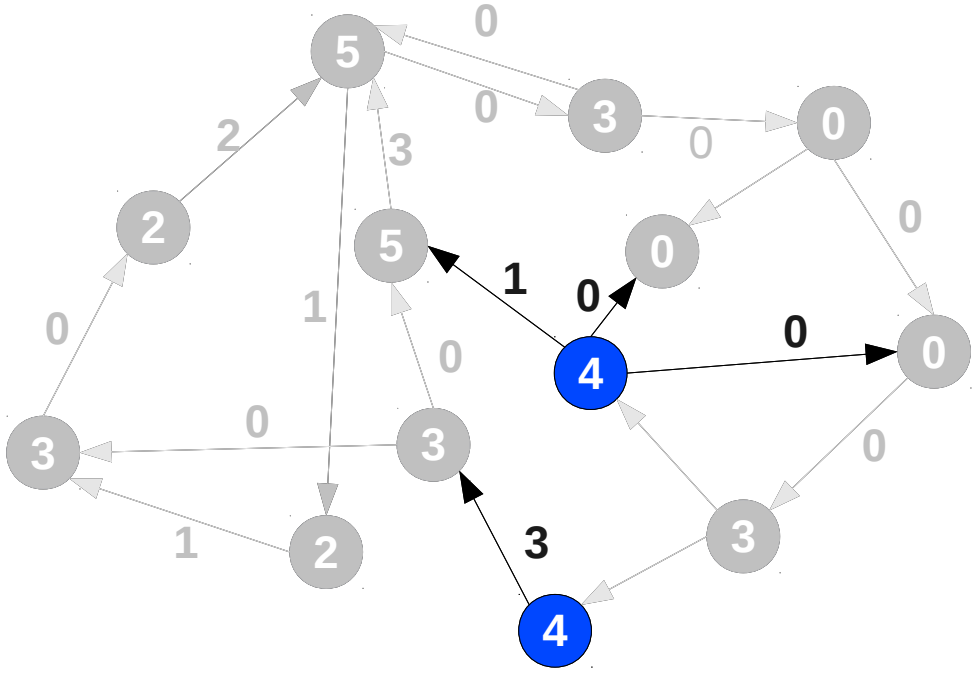
Let (*u*, *v*) be an edge,

α(u) = count of the word *Bhutan* in all the tweets of u mentioning v

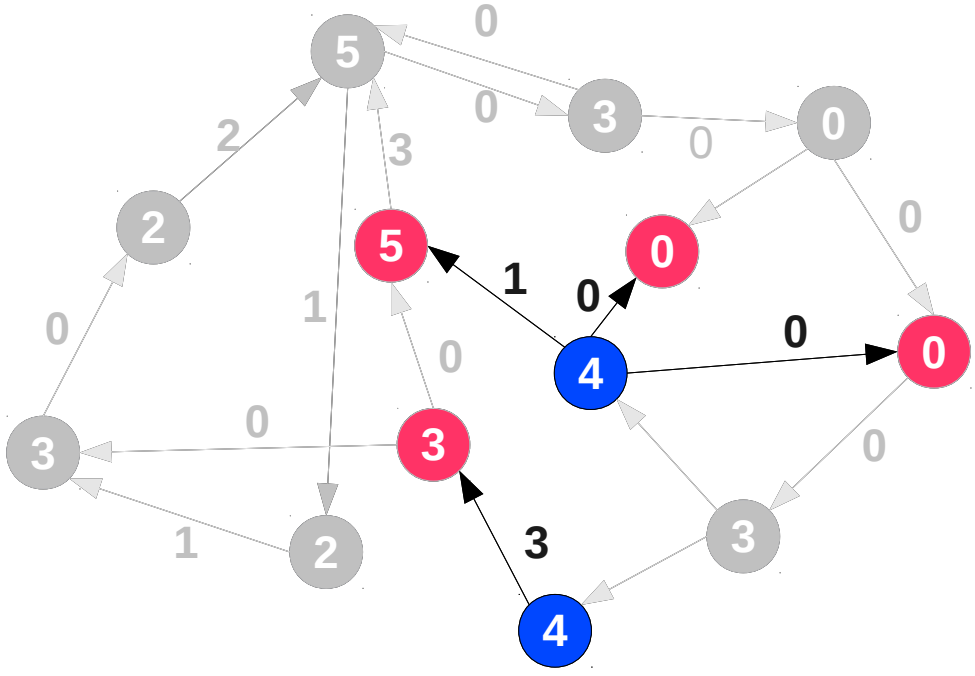
The total graph



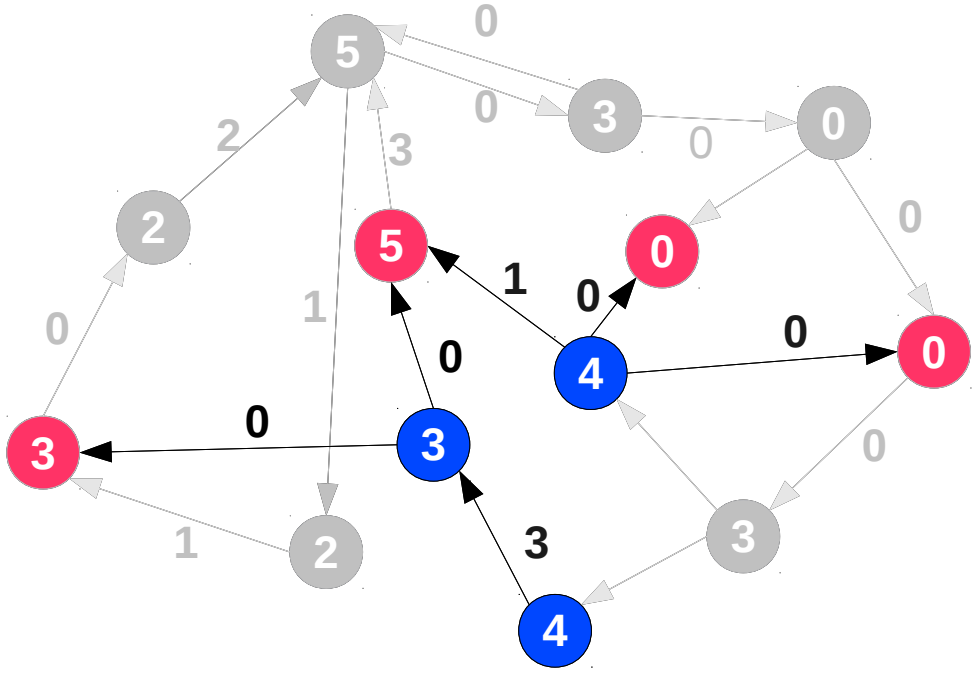
A seed list



The frontier



Crawling one node



A crawl sequence

Let V_0 be the seed list, a set of nodes,
a *crawl sequence,* starting from V_0, is

$$\{ v_i, v_i \text{ in frontier}(V_0 \cup \{v_0, v_1, .. , v_{i-1}\}) \}$$

Goal of a focused crawler

Produce crawl sequences with

global scores (sum) as high as possible

A high-level algorithm

Estimate scores at the frontier

Pick a node from the frontier

Crawl the node

Estimators

- Different estimators can be used:
  - Distance with respect to a seed node
  - Average score of pages pointing to a node
  - Average score of pages pointed to by pages pointing to a node
  - etc.
- Possible to automatically find the estimators best adapted to a given focus crawl using reinforcement learning [Gouriten et al., 2014]

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

Free software

wget	simple yet effective Web spider
Heritrix	Web-scale highly configurable Web crawler, used by the Internet Archive
Beautiful Soup	Python module for parsing real-world Web pages
Scrapy	rich Python module for Web crawling and content extraction
Selenium	browser instrumentor, with API in several languages

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.

Luciano Barbosa and Juliana Freire. Siphoning hidden-Web data through keyword-based interfaces. In *Proc. Simpósio Brasileiro de Bancos de Dados*, Brasília, Brasil, October 2004.

Michael Benedikt, Pierre Bourhis, and Clemens Ley. Querying schemas with access restrictions. *PVLDB*, 5(7), 2012.

BrightPlanet. The deep Web: Surfacing hidden value. White Paper, July 2000.

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.

Bibliography II

Andrea Calì and Davide Martinenghi. Querying Data under Access Limitations. In *ICDE*, 2008.

Andrea Calì and Davide Martinenghi. Querying the deep web. In *EDBT*, 2010. Tutorial.

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.

Kevin Chen-Chuan Chang, Bin He, Chengkai Li, Mitesh Patel, and Zhen Zhang. Structured databases on the Web: Observations and implications. *SIGMOD Record*, 33(3):61–70, September 2004.

Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang. Toward large scale integration: Building a metaquerier over databases on the Web. In *Proc. CIDR*, Asilomar, USA, January 2005.

Bibliography III

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

Muhammad Faheem and Pierre Senellart. Demonstrating intelligent crawling and archiving of web applications. In *Proc. CIKM*, pages 2481–2484, San Francisco, USA, October 2013a. Demonstration.

Muhammad Faheem and Pierre Senellart. Intelligent and adaptive crawling of Web applications for Web archiving. In *Proc. ICWE*, pages 306–322, Aalborg, Denmark, July 2013b.

Muhammad Faheem and Pierre Senellart. Adaptive crawling driven by structure-based link classification, July 2014. Preprint available at `http://pierre.senellart.com/publications/faheem2015adaptive.pdf`.

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

Bibliography IV

Georges Gouriten, Silviu Maniu, and Pierre Senellart. Scalable, generic, and adaptive systems for focused crawling. In *Proc. Hypertext*, Santiago, Chile, September 2014. Douglas Engelbart Best Paper Award.

Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. Accessing the deep Web: A survey. *Communications of the ACM*, 50(2):94–101, 2007.

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

Xin Jin, Nan Zhang, and Gautam Das. Attribute domain discovery for hidden Web databases. In *SIGMOD*, 2011.

Ritu Khare, Yuan An, and Il-Yeol Song. Understanding deep Web search interfaces: a survey. *SIGMOD Record*, 39(1), 2010.

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

Bibliography V

Chen Li and Edward Chang. Answering queries with useful bindings. *ACM TODS*, 26(3), 2001.

Jayant Madhavan, Alon Y. Halevy, Shirley Cohen, Xin Dong, Shawn R. Jeffery, David Ko, and Cong Yu. Structured data meets the Web: A few observations. *IEEE Data Engineering Bulletin*, 29(4):19–26, December 2006.

Richi Nayak, Pierre Senellart, Fabian M. Suchanek, and Aparna Varde. Discovering interesting information with advances in Web technology. *SIGKDD Explorations*, 14(2), December 2012.

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.

Bibliography VI

Pierre Senellart, Avin Mittal, Daniel Muschick, Rémi Gilleron, and Marc Tommasi. Automatic wrapper induction from hidden-Web sources with domain knowledge. In *Proc. WIDM*, pages 9–16, Napa, USA, October 2008.

Cheng Sheng, Nan Zhang, Yufei Tao, and Xin Jin. Optimal algorithms for crawling a hidden database in the Web. *PVLDB*, 5(11), 2012.

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

Mohamed A. Soliman, Ihab F. Ilyas, Davide Martinenghi, and Marco Tagliasacchi. Ranking with uncertain scoring functions: semantics and sensitivity measures. In *SIGMOD*, 2011.

Aparna Varde, Fabian M. Suchanek, Richi Nayak, and Pierre Senellart. Knowledge discovery over the deep Web, semantic Web and XML. In *Proc. DASFAA*, pages 784–788, Brisbane, Australia, April 2009. Tutorial.

Bibliography VII

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

Licence de droits d'usage

Contexte public } avec modifications

Par le téléchargement ou la consultation de ce document, l'utilisateur accepte la licence d'utilisation qui y est attachée, telle que détaillée dans les dispositions suivantes, et s'engage à la respecter intégralement.

La licence confère à l'utilisateur un droit d'usage sur le document consulté ou téléchargé, totalement ou en partie, dans les conditions définies ci-après et à l'exclusion expresse de toute utilisation commerciale.
Le droit d'usage défini par la licence autorise un usage à destination de tout public qui comprend :
– le droit de reproduire tout ou partie du document sur support informatique ou papier,
– le droit de diffuser tout ou partie du document au public sur support papier ou informatique, y compris par la mise à la disposition du public sur un réseau numérique,
– le droit de modifier la forme ou la présentation du document,
– le droit d'intégrer tout ou partie du document dans un document composite et de le diffuser dans ce nouveau document, à condition que :
  – L'auteur soit informé.

Les mentions relatives à la source du document et/ou à son auteur doivent être conservées dans leur intégralité.

Le droit d'usage défini par la licence est personnel et non exclusif.
Tout autre usage que ceux prévus par la licence est soumis à autorisation préalable et expresse de l'auteur : sitepedago@telecom-paristech.fr