

Web search

Web data management and distribution

Serge Abiteboul Ioana Manolescu Philippe Rigaux
Marie-Christine Rousset Pierre Senellart



Web Data Management and Distribution
<http://webdam.inria.fr/textbook>

April 30, 2010

Outline

- 1 The World Wide Web
- 2 Web Crawling
- 3 Web Information Retrieval
- 4 Web Graph Mining
- 5 Conclusion

Internet and the Web

Internet: physical network of computers (or **hosts**)

World Wide Web, Web, WWW: logical collection of **hyperlinked** documents

- **static** and **dynamic**
- **public** Web and **private** Webs
- each document (or **Web page**, or **resource**) identified by a URL

Uniform Resource Locators

`https://www.example.com:443/path/to/doc?name=foo&town=bar#para`
scheme
hostname
port
path
query string
fragment

scheme: way the resource can be accessed; generally **http** or **https**

hostname: **domain name** of a host (cf. DNS); hostname of a website may start with `www.`, but not a rule.

port: **TCP port**; defaults: 80 for `http` and 443 for `https`

path: **logical path** of the document

query string: additional parameters (dynamic documents).

fragment: **subpart** of the document

- Query strings and fragments optional
- Empty path: root of the Web server
- Relative URIs with respect to a **context** (e.g., the URI above):

`/titi` `https://www.example.com/titi`

`tata` `https://www.example.com/path/to/tata`

(X)HTML

- Choice format for **Web pages**
- Dialect of **SGML** (the ancestor of XML), but seldom parsed as is
- HTML 4.01: most common version, **W3C recommendation**
- XHTML 1.0: **XML-ization** of HTML 4.01, minor differences
- **Validation** (cf <http://validator.w3.org/>). Checks the conformity of a Web page with respect to recommendations, for accessibility:
 - ▶ to all graphical browsers (IE, Firefox, Safari, Opera, etc.)
 - ▶ to text browsers (lynx, links, w3m, etc.)
 - ▶ to aural browsers
 - ▶ to all other **user agents** including Web crawlers
- Actual situation of the Web: **tag soup**

XHTML example

```
<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      lang="en" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type"
          content="text/html; charset=utf-8" />
    <title>Example XHTML document</title>
  </head>
  <body>
    <p>This is a
      <a href="http://www.w3.org/">link to the
      <strong>W3C</strong>!</a></p>
  </body>
</html>
```

HTTP

- Client-server **protocol** for the Web, on top of TCP/IP
- Example request/response

```
GET /myResource HTTP/1.1
```

```
Host: www.example.com
```

```
HTTP/1.1 200 OK
```

```
Content-Type: text/html; charset=ISO-8859-1
```

```
<html>
```

```
  <head><title>myResource</title></head>
```

```
  <body><p>Hello world!</p></body>
```

```
</html>
```

- HTTPS: **secure** version of HTTP
 - ▶ encryption
 - ▶ authentication
 - ▶ session tracking

Features of HTTP/1.1

virtual hosting: different Web content for different hostnames on a single machine

login/password protection

content negotiation: same URL identifying several resources, client indicates preferences

cookies: chunks of information persistently stored on the client

keep-alive connexions: several requests using the same TCP connexion
etc.

Outline

- 1 The World Wide Web
- 2 Web Crawling**
- 3 Web Information Retrieval
- 4 Web Graph Mining
- 5 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 index the corresponding page
 - 3 Discover hyperlinks (<a> elements)
 - 4 Repeat on each found link
- 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

Identification of duplicated 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.) identification by hashing of sequences of n successive tokens (**n -grams**)

Crawling ethics

- Standard for robot exclusion: **robots.txt** at the root of a Web server

```
User-agent: *
```

```
Allow: /searchhistory/
```

```
Disallow: /search
```

- Per-page exclusion.

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

- Avoid **Denial Of Service** (DOS), wait 100ms/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`): less overhead
- Use of **keep-alive** to reduce connexion overheads

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:
 - ▶ If-Last-Modified HTTP feature (not reliable)
 - ▶ Identification of **duplicates** in successive request

Outline

- 1 The World Wide Web
- 2 Web Crawling
- 3 Web Information Retrieval**
 - Text Preprocessing
 - Inverted Index
 - Answering Keyword Queries
 - Other Media
- 4 Web Graph Mining
- 5 Conclusion

Information Retrieval, Search

Problem

How to *index* Web content so as to answer (keyword-based) queries *efficiently*?

Context: set of **text documents**

- d_1 The jaguar is a New World mammal of the Felidae family.
- d_2 Jaguar has designed four new engines.
- d_3 For Jaguar, Atari was keen to use a 68K family device.
- d_4 The Jacksonville Jaguars are a professional US football team.
- d_5 Mac OS X Jaguar is available at a price of US \$199 for Apple's new "family pack".
- d_6 One such ruling family to incorporate the jaguar into their name is Jaguar Paw.
- d_7 It is a big cat.

Text Preprocessing

Initial text **preprocessing** steps

- Number of optional steps
- Highly depends on the **application**
- Highly depends on the **document language** (illustrated with English)

Tokenization

Principle

Separate text into **tokens** (words)

Not so easy!

- In some languages (Chinese, Japanese), words **not separated by whitespace**
- Deal **consistently** with acronyms, elisions, numbers, units, URLs, emails, etc.
- **Compound words**: *hostname*, *host-name* and *host name*. Break into two tokens or regroup them as one token? In any case, lexicon and linguistic analysis needed! Even more so in other languages as German.

Usually, remove punctuation and normalize case at this point

Tokenization: Example

- d_1 the₁ jaguar₂ is₃ a₄ new₅ world₆ mammal₇ of₈ the₉ felidae₁₀ family₁₁
- d_2 jaguar₁ has₂ designed₃ four₄ new₅ engines₆
- d_3 for₁ jaguar₂ atari₃ was₄ keen₅ to₆ use₇ a₈ 68k₉ family₁₀ device₁₁
- d_4 the₁ jacksonville₂ jaguars₃ are₄ a₅ professional₆ us₇ football₈ team₉
- d_5 mac₁ os₂ x₃ jaguar₄ is₅ available₆ at₇ a₈ price₉ of₁₀ us₁₁ \$199₁₂
for₁₃ apple's₁₄ new₁₅ family₁₆ pack₁₇
- d_6 one₁ such₂ ruling₃ family₄ to₅ incorporate₆ the₇ jaguar₈ into₉
their₁₀ name₁₁ is₁₂ jaguar₁₃ paw₁₄
- d_7 it₁ is₂ a₃ big₄ cat₅

Stemming

Principle

Merge different forms of the same word, or of closely related words, into a single **stem**

- Not in all applications!
- Useful for retrieving documents containing *geese* when searching for *goose*
- **Various degrees** of stemming
- Possibility of building different indexes, with different stemming

Stemming schemes (1/2)

Morphological stemming.

- Remove **bound morphemes** from words:
 - ▶ plural markers
 - ▶ gender markers
 - ▶ tense or mood inflections
 - ▶ etc.
- Can be linguistically **very complex**, cf:
Les poules du couvent couvent.
[The hens of the monastery brood.]
- In English, somewhat **easy**:
 - ▶ Remove final -s, -'s, -ed, -ing, -er, -est
 - ▶ Take care of semiregular forms (e.g., -y/-ies)
 - ▶ Take care of irregular forms (mouse/mice)
- But still some **ambiguities**: cf stocking

Stemming schemes (2/2)

Lexical stemming.

- Merge **lexically related** terms of various parts of speech, such as *policy*, *politics*, *political* or *politician*
- For English, **Porter's stemming** [Por80]; stem *university* and *universal* to *univers*: not perfect!
- Possibility of coupling this with **lexicons** to merge (near-)synonyms

Phonetic stemming.

- Merge **phonetically related** words: search despite spelling errors!
- For English, **Soundex** [US 07] stems *Robert* and *Rupert* to *R163*. Very **coarse**!

Stemming Example

- d_1 the₁ jaguar₂ be₃ a₄ new₅ world₆ mammal₇ of₈ the₉ felidae₁₀ family₁₁
- d_2 jaguar₁ have₂ design₃ four₄ new₅ engine₆
- d_3 for₁ jaguar₂ atari₃ be₄ keen₅ to₆ use₇ a₈ 68k₉ family₁₀ device₁₁
- d_4 the₁ jacksonville₂ jaguar₃ be₄ a₅ professional₆ us₇ football₈ team₉
- d_5 mac₁ os₂ x₃ jaguar₄ be₅ available₆ at₇ a₈ price₉ of₁₀ us₁₁ \$199₁₂
for₁₃ apple₁₄ new₁₅ family₁₆ pack₁₇
- d_6 one₁ such₂ rule₃ family₄ to₅ incorporate₆ the₇ jaguar₈ into₉
their₁₀ name₁₁ be₁₂ jaguar₁₃ paw₁₄
- d_7 it₁ be₂ a₃ big₄ cat₅

Stop Word Removal

Principle

Remove **uninformative** words from documents, in particular to lower the cost of storing the index

determiners: *a, the, this*, etc.

function verbs: *be, have, make*, etc.

conjunctions: *that, and*, etc.

etc.

Stop Word Removal Example

- d_1 jaguar₂ new₅ world₆ mammal₇ felidae₁₀ family₁₁
- d_2 jaguar₁ design₃ four₄ new₅ engine₆
- d_3 jaguar₂ atari₃ keen₅ 68k₉ family₁₀ device₁₁
- d_4 jacksonville₂ jaguar₃ professional₆ us₇ football₈ team₉
- d_5 mac₁ os₂ x₃ jaguar₄ available₆ price₉ us₁₁ \$199₁₂ apple₁₄
new₁₅ family₁₆ pack₁₇
- d_6 one₁ such₂ rule₃ family₄ incorporate₆ jaguar₈ their₁₀ name₁₁
jaguar₁₃ paw₁₄
- d_7 big₄ cat₅

Inverted Index

After all preprocessing, construction of an **inverted index**:

- Index of **all terms**, with the list of documents where this term **occurs**
- Small scale: disk storage, with **memory mapping** (cf. mmap) techniques; secondary index for offset of each term in main index
- Large scale: distributed on a **cluster of machines**; hashing gives the machine responsible for a given term
- Updating the index costly, so only **batch operations** (not one-by-one addition of term occurrences)

Inverted Index Example

family	d_1, d_3, d_5, d_6
football	d_4
jaguar	$d_1, d_2, d_3, d_4, d_5, d_6$
new	d_1, d_2, d_5
rule	d_6
us	d_4, d_5
world	d_1
...	

Storing positions in the index

- phrase queries, NEAR operator: need to keep **position information** in the index
- just add it in the document list!

family	$d_1/11, d_3/10, d_5/16, d_6/4$
football	$d_4/8$
jaguar	$d_1/2, d_2/1, d_3/2, d_4/3, d_5/4, d_6/8 + 13$
new	$d_1/5, d_2/5, d_5/15$
rule	$d_6/3$
us	$d_4/7, d_5/11$
world	$d_1/6$
...	

TF-IDF Weighting

- Some term occurrences have more **weight** than others:
 - ▶ Terms occurring **frequently** in a **given document**: more **relevant**
 - ▶ Terms occurring **rarely** in the **document collection** as a whole: more **informative**
- Add **Term Frequency—Inverse Document Frequency** weighting to occurrences;

$$\text{tfidf}(t, d) = \frac{n_{t,d}}{\sum_{t'} n_{t',d}} \cdot \log \frac{|D|}{|\{d' \in D \mid n_{t,d'} > 0\}|}$$

$n_{t,d}$ number of occurrences of t in d
 D set of all documents

- Store documents (along with weight) in **decreasing weight order** in the index

TF-IDF Weighting Example

family	$d_1/11/.13, d_3/10/.13, d_6/4/.08, d_5/16/.07$
football	$d_4/8/.47$
jaguar	$d_1/2/.04, d_2/1/.04, d_3/2/.04, d_4/3/.04, d_6/8 + 13/.04, d_5/4/.02$
new	$d_2/5/.24, d_1/5/.20, d_5/15/.10$
rule	$d_6/3/.28$
us	$d_4/7/.30, d_5/11/.15$
world	$d_1/6/.47$
...	

Answering Boolean Queries

- **Single keyword query**: just consult the index and return the documents in index order.
- **Boolean multi-keyword query**

(jaguar AND new AND NOT family) OR cat

Same way! Retrieve document lists from all keywords and apply adequate set operations:

AND intersection

OR union

AND NOT difference

- **Global score**: some function of the individual weight (e.g., addition for conjunctive queries)
- **Position queries**: consult the index, and filter by appropriate condition

Indexing HTML

- HTML: text + meta-information + structure
- Possibly: separate index for meta-information (title, keywords)
- Increase weight of structurally emphasized content in index
- Tree structure can also be queried with XPath or XQuery, but not very useful on the Web as a whole, because of tag soup and lack of consistency.

Indexing Multimedia Content

- Basic approach: index **text from context** of the media
 - ▶ surrounding text
 - ▶ text in or around the links pointing to the content
 - ▶ filenames
 - ▶ associated subtitles (hearing-impaired track on TV)
- Elaborate approach: index and search the media itself, with the help of **speech recognition** and **sound, image, and video analysis**
 - ▶ Musipedia: look for a partition by whistling a tune
 - ▶ Image search from a similar image

Outline

- 1 The World Wide Web
- 2 Web Crawling
- 3 Web Information Retrieval
- 4 Web Graph Mining**
 - PageRank
 - Spamdexing
- 5 Conclusion

The Web Graph

The World Wide Web seen as a (directed) graph:

Vertices: Web pages

Edges: hyperlinks

Same for other interlinked environments:

- dictionaries
- encyclopedias
- scientific publications
- social networks

PageRank (Google's Ranking [BP98])

Idea

Important pages are pages pointed to by **important** pages.

$$\begin{cases} g_{ij} = 0 & \text{if there is no link between page } i \text{ and } j; \\ g_{ij} = \frac{1}{n_i} & \text{otherwise, with } n_i \text{ the number of outgoing links of page } i. \end{cases}$$

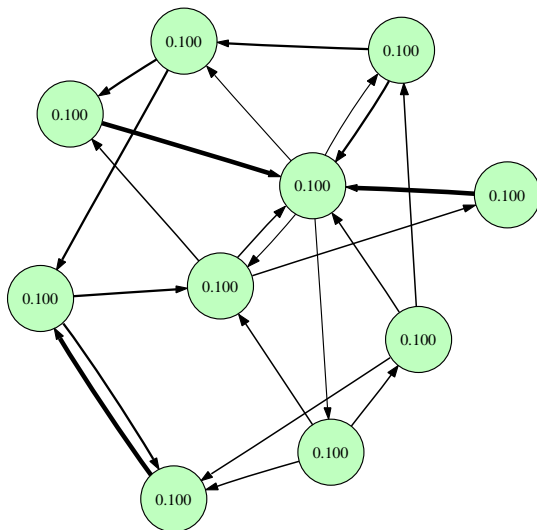
Definition (Tentative)

Probability that the surfer following the **random walk** in G has arrived on page i at some distant given point in the future.

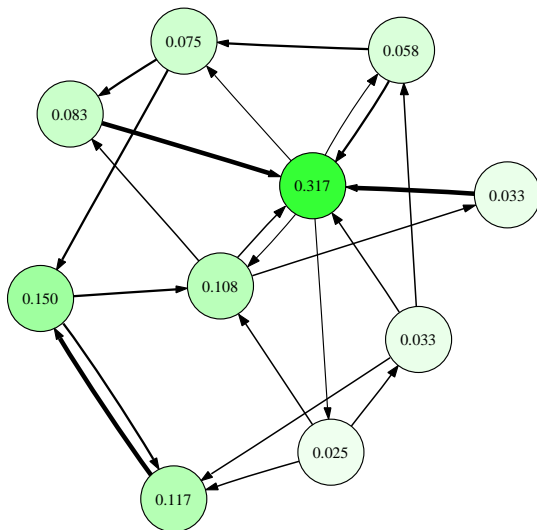
$$\text{pr}(i) = \left(\lim_{k \rightarrow +\infty} (G^T)^k v \right)_i$$

where v is some initial column vector.

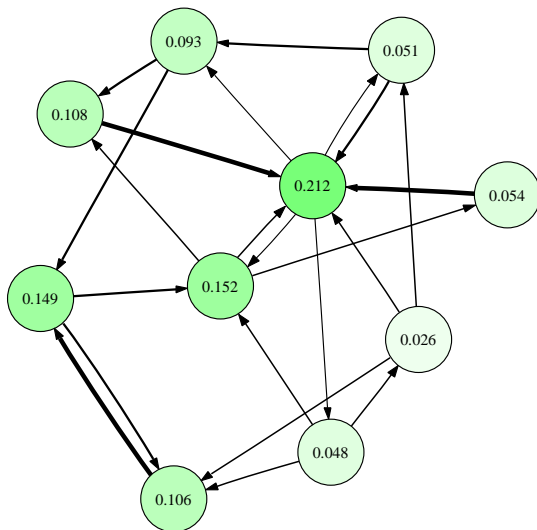
PageRank Iterative Computation



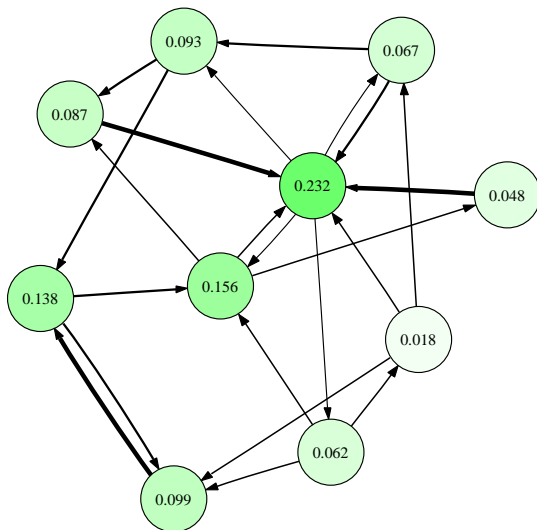
PageRank Iterative Computation



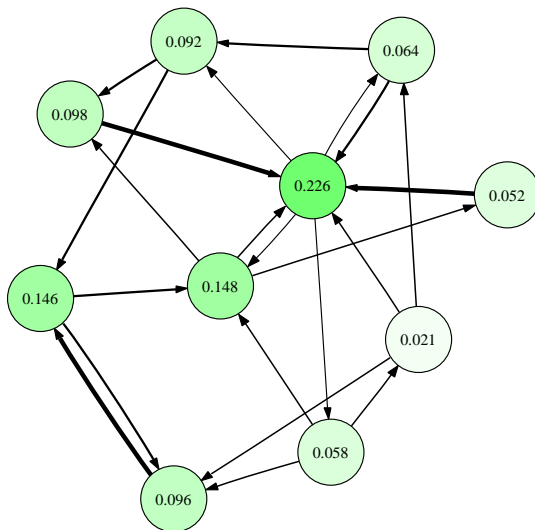
PageRank Iterative Computation



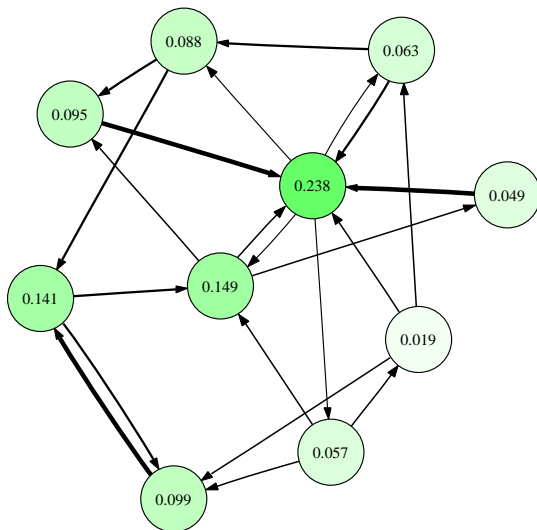
PageRank Iterative Computation



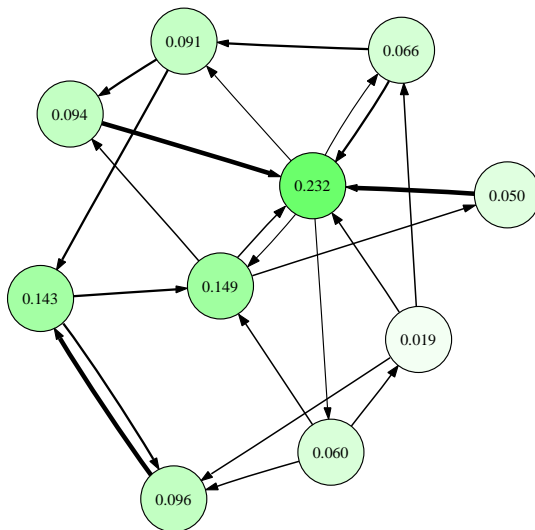
PageRank Iterative Computation



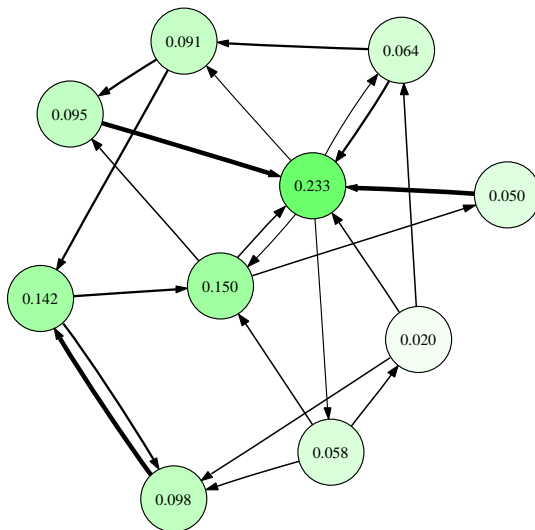
PageRank Iterative Computation



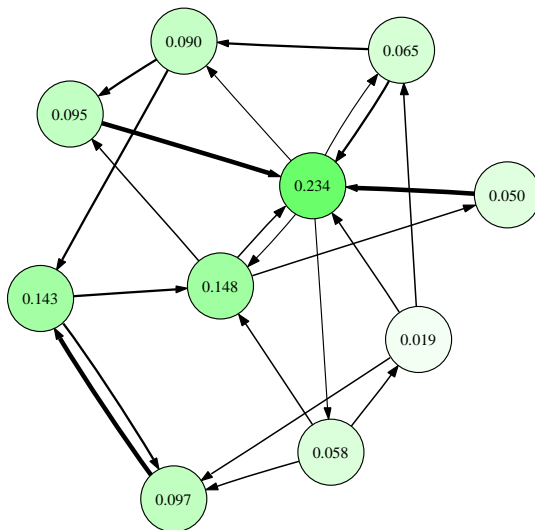
PageRank Iterative Computation



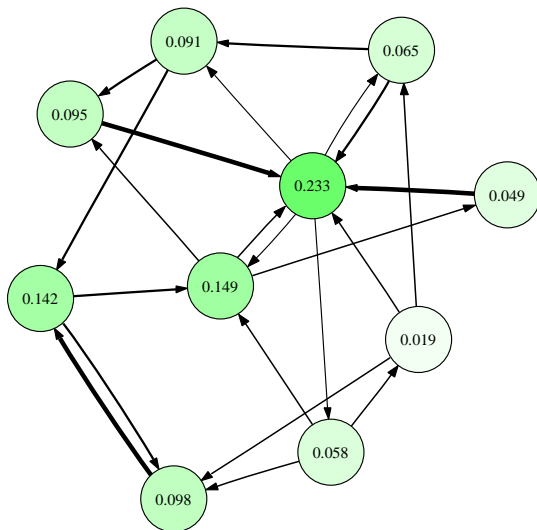
PageRank Iterative Computation



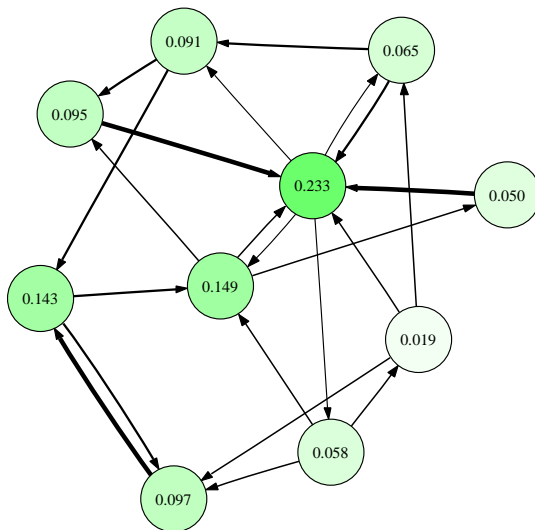
PageRank Iterative Computation



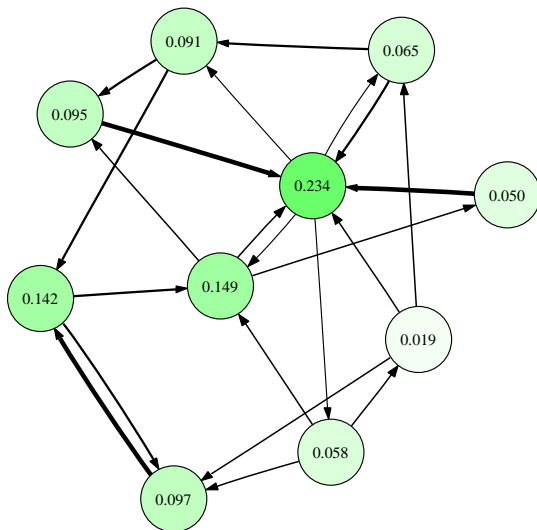
PageRank Iterative Computation



PageRank Iterative Computation



PageRank Iterative Computation



PageRank With Damping

May not always converge, or convergence may not be unique.

To fix this, the random surfer can at each step randomly jump to any page of the Web with some probability d ($1 - d$: damping factor).

$$\text{pr}(i) = \left(\lim_{k \rightarrow +\infty} ((1-d)G^T + dU)^k v \right)_i$$

where U is the matrix with all $\frac{1}{N}$ values with N the number of vertices.

Using PageRank to Score Query Results

- PageRank: **global** score, independent of the query
- Can be used to raise the weight of **important** pages:

$$\text{weight}(t, d) = \text{tfidf}(t, d) \times \text{pr}(d),$$

- This can be directly incorporated **in the index**.

Spamdexing

Definition

Fraudulent techniques that are used by unscrupulous webmasters to artificially raise the visibility of their website to users of search engines

Purpose: attracting visitors to websites to make profit.

Unceasing war between **spamdexers** and **search engines**

Spamdexing: Lying about the Content

Technique

Put **unrelated** terms in:

- meta-information (<meta name="description">, <meta name="keywords">)
- text content hidden to the user with JavaScript, CSS, or HTML presentational elements

Countertechnique

- **Ignore** meta-information
- Try and **detect** invisible text

Link Farm Attacks

Technique

Huge number of hosts on the Internet used for the sole purpose of **referencing** each other, without any content in themselves, to **raise the importance** of a given website or set of websites.

Countertechnique

- Detection of websites with **empty** or **duplicate** content
- Use of heuristics to discover **subgraphs** that look like link farms

Link Pollution

Technique

Pollute **user-editable** websites (blogs, wikis) or exploit security bugs to add **artificial** links to websites, in order to raise its importance.

Countertechnique

rel="nofollow" attribute to `<a>` links not validated by a page's owner

Outline

- 1 The World Wide Web
- 2 Web Crawling
- 3 Web Information Retrieval
- 4 Web Graph Mining
- 5 Conclusion**

What you should remember

- The **inverted index** model for efficient answers of keyword-based queries.
- **PageRank** and its iterative computation.

References





- Specifications

- ▶ HTML 4.01, <http://www.w3.org/TR/REC-html40/>
- ▶ HTTP/1.1, <http://tools.ietf.org/html/rfc2616>
- ▶ *Robot Exclusion Protocol*, <http://www.robotstxt.org/orig.html>

- A book

Mining the Web: Discovering Knowledge from Hypertext Data,
Soumen Chakrabarti, Morgan Kaufmann

Bibliography I

-  Sergey Brin and Lawrence Page.
The anatomy of a large-scale hypertextual Web search engine.
Computer Networks, 30(1–7):107–117, April 1998.
-  Ronald Fagin, Amnon Lotem, and Moni Naor.
Optimal aggregation algorithms for middleware.
Journal of Computer and System Sciences, 66:614–656, 2003.
Abstract published in PODS'2001.
-  Jon M. Kleinberg.
Authoritative Sources in a Hyperlinked Environment.
Journal of the ACM, 46(5):604–632, 1999.
-  Martin F. Porter.
An algorithm for suffix stripping.
Program, 14(3):130–137, July 1980.

Bibliography II



US National Archives and Records Administration.

The Soundex indexing system.

<http://www.archives.gov/genealogy/census/soundex.html>,
May 2007.