Web search

Web data management and distribution

Serge Abiteboul  Philippe Rigaux
Marie-Christine Rousset  Pierre Senellart

http://gemo.futurs.inria.fr/wdmd

November 29, 2009
Outline

1. The World Wide Web
2. Web Crawling
3. Web Information Retrieval
4. Web Graph Mining
5. Hot Topics
6. 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**: 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 optionals
- 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</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 negociaitio:** same URL identifying several resources, client indicates preferences

**cookies:** chunks of information persistently stored on the client

**keep-alive connexion:** 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. Hot Topics
6. 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
   - Clustering
   - Other Media
4. Web Graph Mining
5. Hot Topics
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 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 $199.
d_6 one such ruling family to incorporate the jaguar into their name is jaguar paw.
d_7 it is a big cat.
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 (1/2)

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

\[ d_1 \text{ the} \text{jaguar} \text{ be} \text{ a} \text{ new} \text{ world} \text{ mammal} \text{ of} \text{ the} \text{ felidae} \text{ family} \]
\[ d_2 \text{ jaguar} \text{ have} \text{ design} \text{ four} \text{ new} \text{ engine} \]
\[ d_3 \text{ for} \text{jaguar} \text{ atari} \text{ be} \text{ keen} \text{ to} \text{ use} \text{ a} \text{ 68k} \text{ family} \text{ device} \]
\[ d_4 \text{ the} \text{jacksonville} \text{ jaguar} \text{ be} \text{ a} \text{ professional} \text{ us} \text{ football} \text{ team} \]
\[ d_5 \text{ mac} \text{ os} \text{x} \text{jaguar} \text{ be} \text{ available} \text{ at} \text{ a} \text{ price} \text{ of} \text{ us} \$199 \text{ for} \text{ apple} \text{ new} \text{ family} \text{ pack} \]
\[ d_6 \text{ one} \text{ such} \text{ rule} \text{ family} \text{ to} \text{ incorporate} \text{ the} \text{jaguar} \text{ into} \text{ their} \text{ name} \text{ be} \text{ j jaguar} \text{ paw} \]
\[ d_7 \text{ it} \text{ be} \text{ a} \text{ big} \text{ cat} \]
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

\[d_1\] jaguar_2\ new_5\ world_6\ mammal_7\ felidae_10\ family_{11} \\
\[d_2\] jaguar_1\ design_3\ four_4\ new_5\ engine_6 \\
\[d_3\] jaguar_2\ atari_3\ keen_5\ 68k_9\ family_{10}\ device_{11} \\
\[d_4\] jacksonville_2\ jaguar_3\ professional_6\ us_7\ football_8\ team_9 \\
\[d_5\] mac_1\ os_2\ x_3\ jaguar_4\ available_6\ price_9\ us_11\ $199_{12}\ apple_{14} \\
\] new_15\ family_{16}\ pack_{17} \\
\[d_6\] one_1\ such_2\ rule_3\ family_4\ incorporate_6\ jaguar_8\ their_{10}\ name_{11} \\
\] jaguar_{13}\ paw_{14} \\
\[d_7\] big_4\ cat_5
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

<table>
<thead>
<tr>
<th>Term</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>family</td>
<td>$d_1, d_3, d_5, d_6$</td>
</tr>
<tr>
<td>football</td>
<td>$d_4$</td>
</tr>
<tr>
<td>jaguar</td>
<td>$d_1, d_2, d_3, d_4, d_5, d_6$</td>
</tr>
<tr>
<td>new</td>
<td>$d_1, d_2, d_5$</td>
</tr>
<tr>
<td>rule</td>
<td>$d_6$</td>
</tr>
<tr>
<td>us</td>
<td>$d_4, d_5$</td>
</tr>
<tr>
<td>world</td>
<td>$d_1$</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Storing positions in the index

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

<table>
<thead>
<tr>
<th>Term</th>
<th>Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>family</td>
<td>$d_1/11$, $d_3/10$, $d_5/16$, $d_6/4$</td>
</tr>
<tr>
<td>football</td>
<td>$d_4/8$</td>
</tr>
<tr>
<td>jaguar</td>
<td>$d_1/2$, $d_2/1$, $d_3/2$, $d_4/3$, $d_5/4$, $d_6/8 + 13$</td>
</tr>
<tr>
<td>new</td>
<td>$d_1/5$, $d_2/5$, $d_5/15$</td>
</tr>
<tr>
<td>rule</td>
<td>$d_6/3$</td>
</tr>
<tr>
<td>us</td>
<td>$d_4/7$, $d_5/11$</td>
</tr>
<tr>
<td>world</td>
<td>$d_1/6$</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
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;

\[
tfidf(t, d) = \frac{n_{t,d}}{\sum_{t'} n_{t',d}} \cdot \log \left( \frac{|D|}{|\{d' \in D \mid n_{t,d'} > 0\}|} \right)
\]

- \( n_{t,d} \) 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**

\[(\text{jaguar AND new AND NOT family}) \text{ OR cat}\]

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
Answering Top-\(k\) Queries

\[ t_1 \text{ AND } \ldots \text{ AND } t_n \]

**Problem**

Find the **top-\(k\) results** (for some given \(k\)) to the query, without retrieving all documents matching it.

**Notations:**

\[ s(t, d) \quad \text{weight of } t \text{ in } d \text{ (e.g., tfidf)} \]

\[ g(s_1, \ldots, s_n) \quad \text{monotonous function that computes the global score (e.g., addition)} \]
Fagin’s Threshold Algorithm [FLN03]

1. Let $R$ be the empty list, and $m = +\infty$.
2. For each $1 \leq i \leq n$:
   1. Retrieve the document $d^{(i)}$ containing term $t_i$ that has the next largest $s(t_i, d^{(i)})$.
   2. Compute its global score $g_{d^{(i)}} = g(s(t_1, d^{(i)}), \ldots, s(t_n, d^{(i)}))$ by retrieving all $s(t_j, d^{(i)})$ with $j \neq i$.
   3. If $R$ contains less than $k$ documents, or if $g_{d^{(i)}}$ is greater than the minimum of the score of documents in $R$, add $d^{(i)}$ to $R$.
3. Let $m = g(s(t_1, d^{(1)}), s(t_2, d^{(2)}), \ldots, s(t_n, d^{(n)}))$.
4. If $R$ contains more than $k$ documents, and the minimum of the score of the documents in $R$ is greater than or equal to $m$, return $R$.
5. Redo step 2.
Clustering Example

Top 232 results of at least 13,030,000 retrieved for the query jaguar (definition) (details)

1. **jaguars.com -- The official web site of the NFL's Jacksonville Jaguars**
   The official team site with scores, news items, game schedule, and roster.

2. **Jaguar**
   The jaguar (Panthera onca) is a large member of the cat family native to warm
   regions of the Americas. It is closely related to the lion, tiger, and leopard of the Old
   World, and is the largest species of the cat family found in the Americas.

3. **Jaguar Enthusiasts' Club**
   World's largest Jaguar / Daimler Club ... Largest Jaguar Club in the World, serving over 20,000
   members ...
   www.je.co.uk - [cache] - Ask, Open Directory

4. **US abandons bid for jaguar recovery plan**
   Jan 18, 2008 - The Interior Department has abandoned attempts to craft a recovery plan for the
   endangered jaguar because too few of the rare cats have been spotted along the Southwest region
   of New Mexico and Arizona to warrant such action. Some critics of the decision said Thursday the
   jaguar is being sacrificed for the government’s new border fence, which is going up along many of the
   same areas where the ... has crossed into the United States from Mexico. If the U.S. border areas
Cosine Similarity of Documents

- **Document Vector Space model:**
  - terms dimensions
  - documents vectors
  - coordinates weights

  (The projection of document $d$ along coordinate $t$ is the weight of $t$ in $d$, say $\text{tfidf}(t, d)$)

- Similarity between documents $d$ and $d'$: *cosine* of these two vectors

  \[
  \cos(d, d') = \frac{d \cdot d'}{\|d\| \times \|d'\|}
  \]

  - $d \cdot d'$ scalar product of $d$ and $d'$
  - $\|d\|$ norm of vector $d$

- $\cos(d, d) = 1$

- $\cos(d, d') = 0$ if $d$ and $d'$ are *orthogonal* (do not share any common term)
Agglomerative Clustering of Documents

1. Initially, each document forms its own cluster.

2. The similarity between two clusters is defined as the maximal similarity between elements of each cluster.

3. Find the two clusters whose mutual similarity is highest. If it is lower than a given threshold, end the clustering. Otherwise, regroup these clusters. Repeat.

Remark

Many other more refined algorithms for clustering exist.
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
   - HITS
   - Spamdexing
5. Hot Topics
6. 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.

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

**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.

\[
pr(i) = \left( \lim_{k \to +\infty} (G^T)^k \nu \right)_i
\]

where \( \nu \) 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 Iterative Computation

The diagram illustrates the iterative computation of PageRank values for a web graph. Each node represents a webpage, and the numbers indicate the PageRank value assigned to each node. The arrows represent hyperlinks between pages, and the PageRank values are updated iteratively based on the PageRank algorithm, which considers both the number of incoming links and their importance.
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).

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

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.
HITS (Kleinberg, [Kle99])

Idea

Two kinds of important pages: hubs and authorities. Hubs are pages that point to good authorities, whereas authorities are pages that are pointed to by good hubs.

$G'$ transition matrix (with 0 and 1 values) of a subgraph of the Web. We use the following iterative process (starting with $a$ and $h$ vectors of norm 1):

$$\begin{cases} 
    a := \frac{1}{\|G'h\|} G'T h \\
    h := \frac{1}{\|G'a\|} G'a
\end{cases}$$

Converges under some technical assumptions to authority and hub scores.
Using HITS to Order Web Query Results

1. Retrieve the set $D$ of Web pages matching a keyword query.
2. Retrieve the set $D^*$ of Web pages obtained from $D$ by adding all linked pages, as well as all pages linking to pages of $D$.
3. Build from $D^*$ the corresponding subgraph $G'$ of the Web graph.
4. Compute iteratively hubs and authority scores.
5. Sort documents from $D$ by authority scores.

Less efficient than PageRank, because local scores.
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 (\texttt{<meta name="description">}, \texttt{<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. Hot Topics
   - Semantic Web
   - Web 2.0
   - Deep Web

6. Conclusion
Querying the Semantic Web

Definition

**Semantic Web**: extension of the current Web, where human-readable content is annotated with machine-readable descriptions

- **RDF** to describe objects, and graphs of relationships between objects
- **RDFS** and **OWL** to express schemata and ontologies
- **SPARQL** to query semantic Web sources
- **Problem**: no uniformity in schemata and ontologies on the Web
  \[ \Rightarrow \] integration needed
Definition

Web 2.0: buzzword about:

- rich dynamic interfaces, especially with the help of AJAX (Asynchronous JavaScript and XML) technologies: GMail, Google Suggest
- user-editable content, collaborative work and social networks: blogs, Wikipedia, MySpace, Facebook
- aggregation of content from multiple sources and personalization: Netvibes, Yahoo! Pipes

Interesting issues:

- application of graph mining techniques to the graph of social network websites
- mashups for aggregating content from multiple sources on the Web
The Deep Web

**Definition**

**Deep Web** (or hidden Web, or invisible Web): part of Web content that lies in online databases, typically queried through HTML forms, and is not usually accessible by following hyperlinks.

- **Huge** amount of information (maybe 500 more than on the **surface Web**?): *Yellow pages* directories, information from the US *Census bureau*, weather or geolocation services.
- **Extensional** (siphoning) or **intensional** (understanding services) approaches.
Outline

1. The World Wide Web
2. Web Crawling
3. Web Information Retrieval
4. Web Graph Mining
5. Hot Topics
6. Conclusion
What you should remember

- The inverted index model for efficient answers of keyword-based queries.
- The threshold algorithm for retrieving top-\( k \) results.
- PageRank and its iterative computation.
References

Specifications
- HTML 4.01, http://www.w3.org/TR/REC-html40/

A book

*Mining the Web: Discovering Knowledge from Hypertext Data*, Soumen Chakrabarti, Morgan Kaufmann
Sergey Brin and Lawrence Page.  
The anatomy of a large-scale hypertextual Web search engine.  

Ronald Fagin, Amnon Lotem, and Moni Naor.  
Optimal aggregation algorithms for middleware.  

Jon M. Kleinberg.  
Authoritative Sources in a Hyperlinked Environment.  

Martin F. Porter.  
An algorithm for suffix stripping.  