

# Web Information Retrieval

M2R – MOSIG  
2016-2017

Philippe Mulhem

Philippe.Mulhem@imag.fr

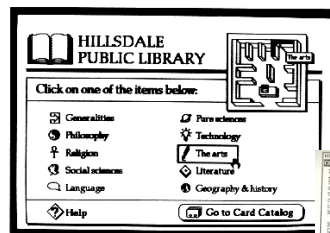
## Outline

- The Web
  - Overview
  - Web as graph
- Web document access
  - Crawler/Indexing/Content Retrieval
  - Graph Usage
  - Integration of elements for Web retrieval
- Conclusion

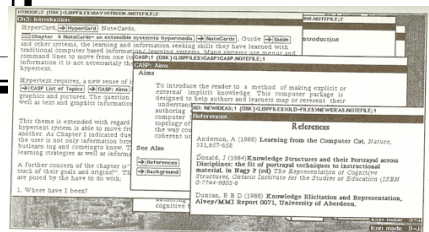
# The Web - Overview

- ... Before 1989
  - Internet Protocol (IP)
  - File transfert Protocol (FTP)
  - Discussions forums (Usenet), mails (SMTP)
  - Numerous file formats (PS, DVI, TXT, PDF, ...)
  - Hypertext softwares

- hypecard/Apple →



- Notecards/Xerox →



Web IR - MOSIG

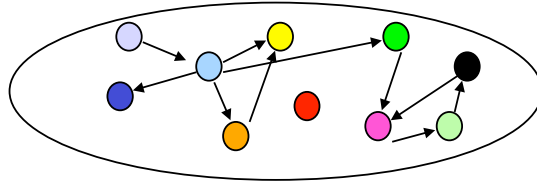
# The Web - Overview

- Tim Berners-Lee: bring all these technologies together
  - Unique resource description : URL
  - Simple Hypertext description for URL as links : HTML
  - A new simple, full text and stateless protocol on IP : HTTP

- Now, largest source of data and information

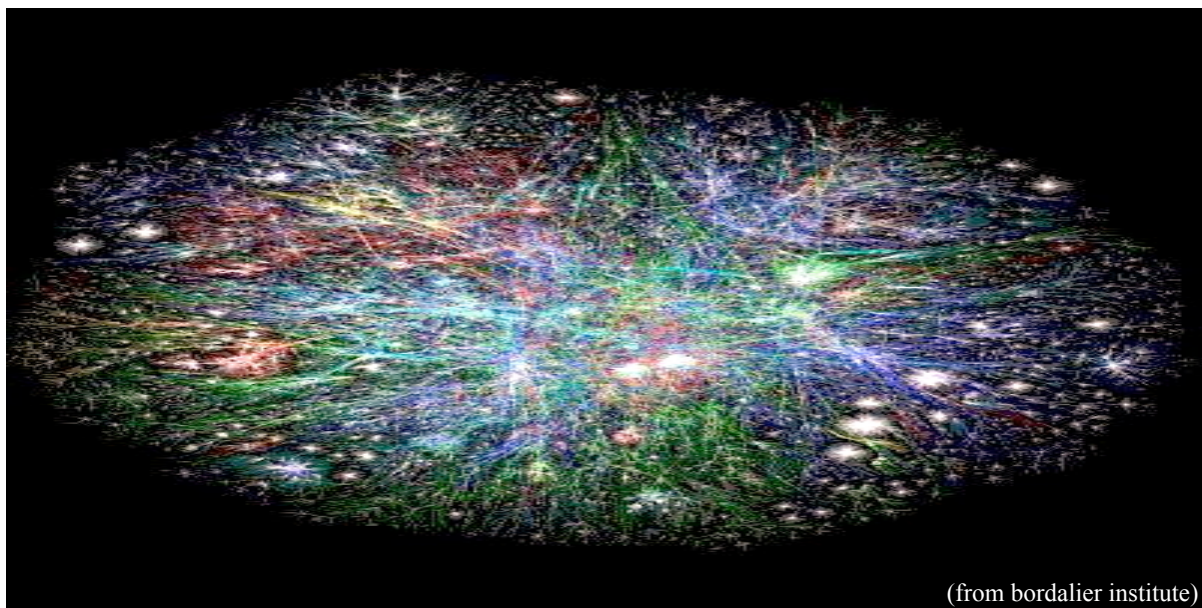
# The Web as a graph

- Web pages: nodes
- Hypertext links: edges (meanings: structure, recommendation, similarity, reading paths ?)



- Then graph processing may be used (see later)

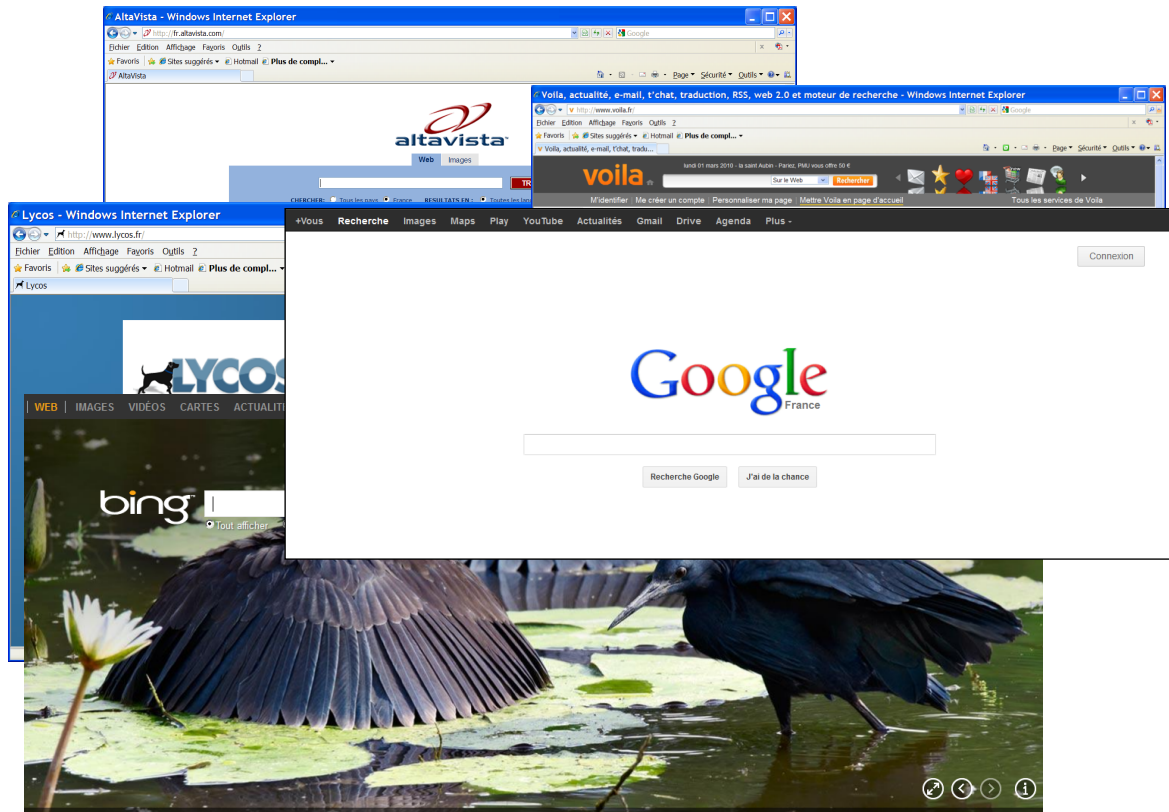
## « The Web » is a Huge Graph



(from bordalier institute)

- With
  - Static pages (surface web, above)
  - Dynamic pages (deep web)

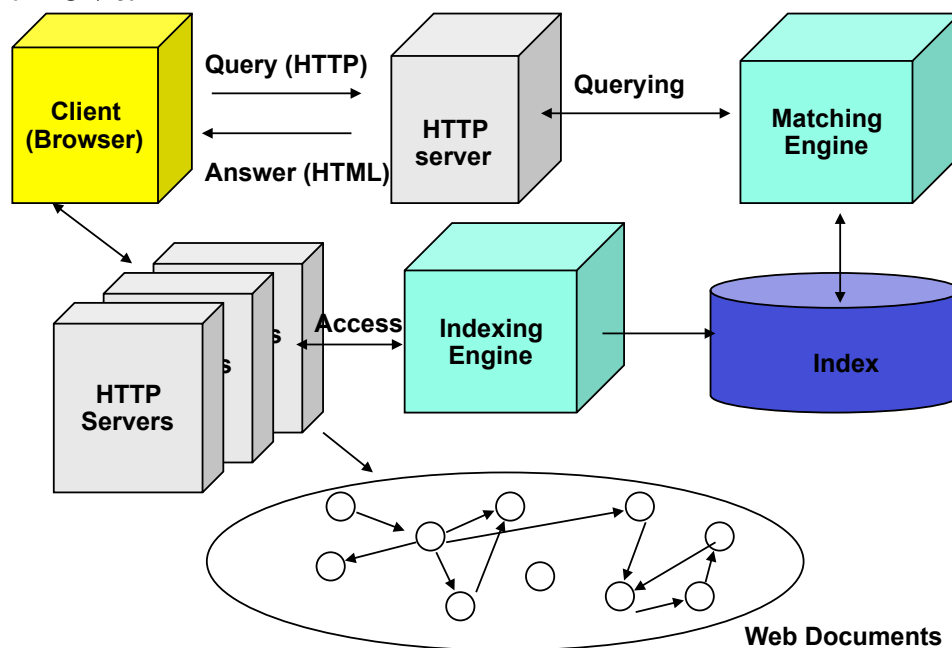
# Web search interfaces



Web IR - MOSIG

# Web documents access

- Information available but not accessible... need for retrieval



Web IR - MOSIG

# Web Retrieval by queries

- General features
  - Automatic indexing
    - Robust: heterogeneity (languages, documents structures & formats)
  - Coverage (Google : 45 billions of pages)
    - Huge storage capacity (hundreds of TeraBytes)
  - Freshness
  - Response time (for interaction, 4 billions queries/day)
    - Multi-level indexing (content and structure)
    - Huge processing
    - Models that mix content/structure/links (boolean models with words position, query logs, social data)

## Indexing engine - Crawler

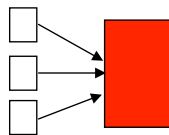
- Dynamic exploration of the web (bots)
  - Googlebot, Bingbot, Slurp (Yahoo), DuckDuckBot
- Simple crawler using a set of seeds URL named E
  - **while** E not empty **do**
    - $e \leftarrow \text{get\_one\_element\_of}(E)$
    - $p \leftarrow \text{get\_page\_from\_URL}(e)$
    - index(p)
    - $E = E \cup \text{outlinks}(p)$
  - Refinements on
    - Number of pages, Topics
    - Efficiency (wrt. overload of web servers)
- Classical inverted indexes with term position, adapted to high dimensions (terms x documents x queries)
  - Distributed indexes

# Content Retrieval

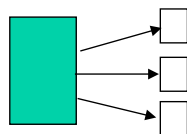
- Queries on the web are generally short (2-4 words)
- « The best place to hide a dead body is page 2 of the Google search results. » (Brian Clark)
  - people are interested in the first result page
    - All elements that may enhance the top-10 results are used (web server features, social tags, HTML validations, ...)
- Classical Boolean models with term position
  - AND/OR/NOT
  - Proximity search
- 100's of other parameters (not only content)

## Graph Usage – HITS [Kleinberg99]

- Goal: for a query (topic), to find
  - Good sources of content (Authorities, AUTH)



- Good sources of links (Hubs, HUB)

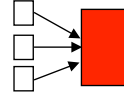


# Graph Usage – HITS [Kleinberg99]

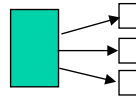
- Pages characterized by <HUB; AUTH> value

## Intuition

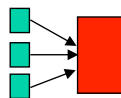
- Authority value from in-links



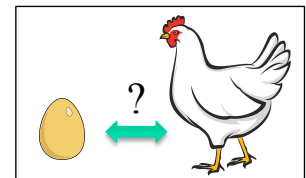
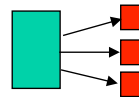
- Hub value from out-links



- High authority if in-links from good Hubs



- High Hub if out-links to good Authorities



# Graph Usage – HITS [Kleinberg99]

- Initialisation :

$$\sum AUTH[U]^2 = \sum HUB[U]^2 = 1 \quad (\text{if } N \text{ pages: } AUTH[V] = HUB[V] = \frac{1}{\sqrt{N}})$$

- Iteration until convergence

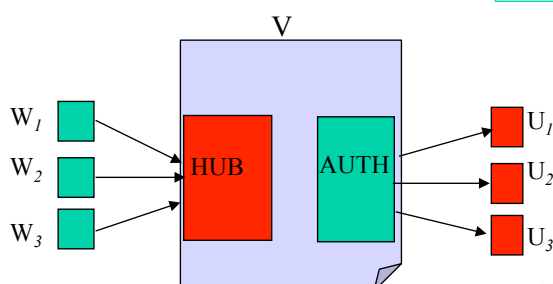
$$1. \text{ For each } V : AUTH'[V] = \sum_{Edge(W_i, V)} HUB[W_i]$$

$$HUB'[V] = \sum_{Edge(V, U_i)} AUTH[U_i]$$

- 2. Normalization:

$$AUTH[V] = \frac{AUTH'[V]}{\sqrt{\sum_U AUTH'[U]^2}}$$

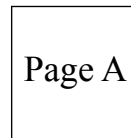
$$HUB[V] = \frac{HUB'[V]}{\sqrt{\sum_U HUB'[U]^2}}$$



$$\text{so that } \sum AUTH[U]^2 = \sum HUB[U]^2 = 1$$

# Graph Usage – HITS [Kleinberg99]

- Example



- $AUTH[A]=HUB[A]=1$

# Graph Usage – HITS [Kleinberg99]

- Example

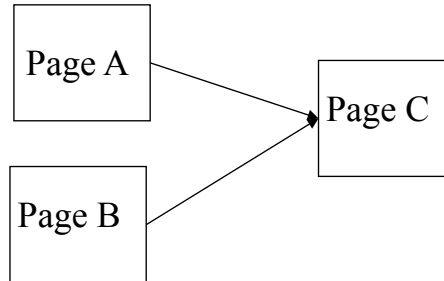


	AUTH[A]	HUB[A]	AUTH[B]	HUB[B]	$\sqrt{\sum AUTH^2[U]^2}$	$\sqrt{\sum HUB^2[U]^2}$
0	0.71	0.71	0.71	0.71		
1.1	0	0.71	0.71	0	0.71	0.71
1.2	0	1	1	0		
2.1	0	1	1	0	1	1
2.2	0	1	1	0		



# Graph Usage – HITS [Kleinberg99]

- Example

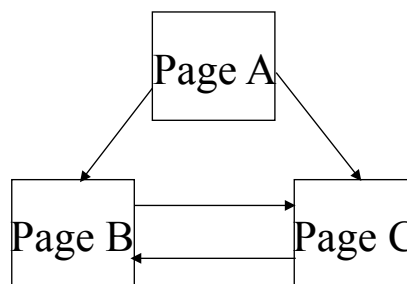


	AUTH[A]	HUB[A]	AUTH[B]	HUB[B]	AUTH[C]	HUB[C]	$\sqrt{\sum A[U]^2}$	$\sqrt{\sum H[U]^2}$
0	0.58	0.58	0.58	0.58	0.58	0.58		
1.1	0	0.58	0	0.58	1.15	0	1.15	0.82
1.2	0	0.71	0	0.71	1	0		
2.1	0	1	0	1	1.41	0	1.41	1.41
2.2	0	0.71	0	0.71	1	0		

Web IR - MOSIG

# Graph Usage – HITS [Kleinberg99]

- Example

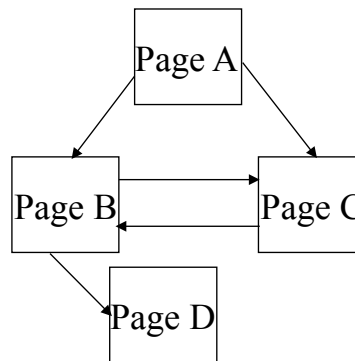


	AUTH[A]	HUB[A]	AUTH[B]	HUB[B]	AUTH[C]	HUB[C]
0	0.58	0.58	0.58	0.58	0.58	0.58
1.1	0	1,15	1,15	0,58	1,15	0,58
1.2	0	0,82	0,71	0,41	0,71	0,41
2.1	0	1,41	1,22	0,71	1,22	0,71
2.2	0	0,82	0,71	0,41	0,71	0,41

# Graph Usage – HITS [Kleinberg99]

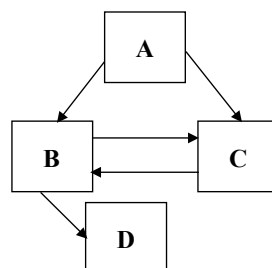
- Example

Note : stop when mean of differences  $\leq 0.02$



	AUTH[A]	HUB[A]	AUTH[B]	HUB[B]	AUTH[C]	HUB[C]	AUTH[D]	HUB[D]
0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
1.1								
1.2								
2.1								
2.2								

...



AUTH[A]	HUB[A]	AUTH[B]	HUB[B]	AUTH[C]	HUB[C]	AUTH[D]	HUB[D]
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0,00	1,00	1,00	1,00	1,00	0,50	0,50	0,00
0,00	0,67	0,67	0,67	0,67	0,33	0,33	0,00
0,00	1,33	1,00	1,00	1,33	0,67	0,67	0,00
0,00	0,74	0,56	0,56	0,74	0,37	0,37	0,00
0,00	1,30	1,11	1,11	1,30	0,56	0,56	0,00
0,00	0,72	0,62	0,62	0,72	0,31	0,31	0,00
0,00	1,34	1,03	1,03	1,34	0,62	0,62	0,00
0,00	0,74	0,57	0,57	0,74	0,34	0,34	0,00
0,00	1,32	1,09	1,09	1,32	0,57	0,57	0,00
0,00	0,73	0,60	0,60	0,73	0,32	0,32	0,00

# Graph Usage – HITS [Kleinberg99]

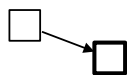
- Conclusion
  - Overload for each query processed
  - « Heavy » computation
  - Easy to spam with automatically generated links
- Is there others ways to avoid such drawbacks ?

# Graph Usage – Pagerank

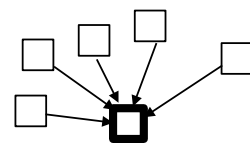
- Google [Brin & Page 1998]
  - Pagerank computes the popularity of web pages
    - +++ independently of any query
  - A Pagerank value is one positive floating point value
    - +++ simple
  - Graphically:



Low popularity



Medium popularity



High popularity

## Graph Usage – Pagerank

- All web pages vote for the popularity of one Web page.
- An in-link on a page is voting « for » the page.
- No link is an « abstention » about the page
- Pagerank computes the « chances » to access one page A from any other page on the web, according A's in-links

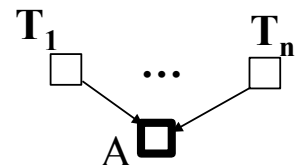
## Graph Usage – Pagerank

- The Pagerank, PR, of a page A is defined as:

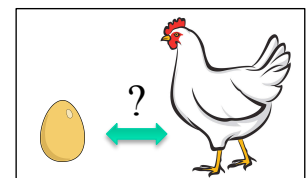
$$PR(A) = (1 - d) + d * \left( \frac{PR(T_1)}{C(T_1)} + \dots + \frac{PR(T_n)}{C(T_n)} \right)$$

avec

- d : dumping factor in  $[0,1]$
- $T_i$  : Web that link to A
- $C(T_i)$  : number of out-links from  $T_i$
- PR in  $[1-d, +\infty[$

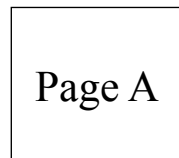


- How to compute PR(A)???
- Iteration until convergence, with initial values of 1.0



## Graph Usage – Pagerank

- Example



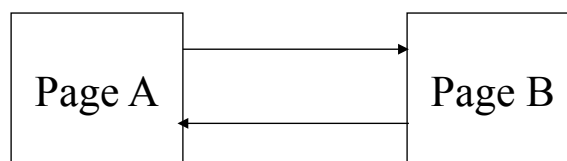
–  $d=0.85$  et  $\text{init PR}(A) = 1$

–  $\text{PR}(A)=(1-d) = 0.15$

## Graph Usage – Pagerank

- Example

–



–  $d=0.85$ ,  $\text{init PR}(A) = \text{PR}(B) = 1$

–  $\text{PR}(A)=(1-d)+d(\text{PR}(B)/1) = 0.15+0.85*1 = 1$

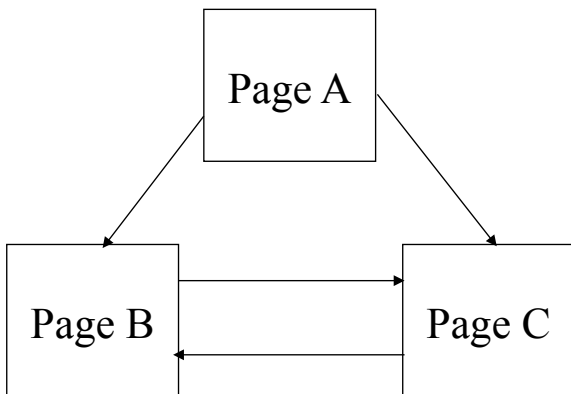
–  $\text{PR}(B)=(1-d)+d(\text{PR}(A)/1) = 0.15+0.85*1 = 1$

– Initial values do not change in this case.

# Graph Usage – Pagerank

- Example

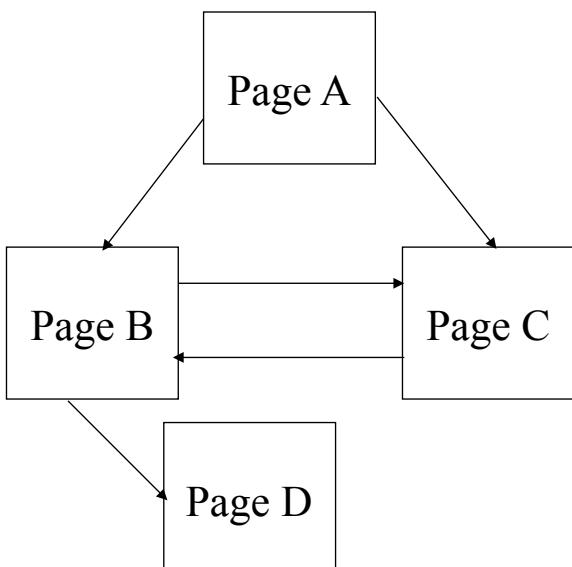
–  $d=0.85$ , init  $PR(A) = PR(B) = PR(C)=1$



	PR(A)	PR(B)	PR(C)
1	1	1	1
2	0,15	1,425	1,425
3	0,15	1,425	1,425

# Graph Usage – Pagerank

- Example

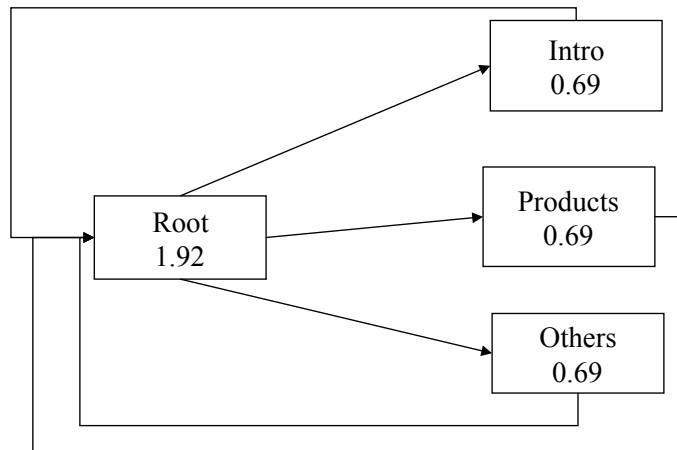


	PR(A)	PR(B)	PR(C)	PR(D)
1	1	1	1	1
2	0.15	1.425	1	0.575
3	0,15	1,06	0,82	0,76
4	0,15	0,91	0,67	0,60
5	0,15	0,78	0,60	0,54
6	0,15	0,72	0,55	0,48
7	0,15	0,68	0,52	0,46
8	0,15	0,66	0,50	0,44
9				

Note : stop when average of differences lower that threshold 0.02.

# Graph Usage – Pagerank

- **Example** ([www.iprcom.com/papers/pagerank](http://www.iprcom.com/papers/pagerank) plus actif)
  - Simple hierarchy



Web IR - MOSIG

29

# Graph Usage – Pagerank

- **Conclusion**
  - + A priori computation
  - + Hard to spam
  - Complex to compute on billions of pages
  - Disadvantages novelty
  - Choice of one interpretation of links

Web IR - MOSIG

30

# Integration of elements for Web retrieval

- Integration with content-based matching using learning to rank

- Letor 3.0 GOV corpus

[Qin, Liu, Xu, Li 2010]

- 64 features
- 575 queries
- Features of top 1000 docs

21	BM25 of body	Q-D
22	BM25 of anchor	Q-D
23	BM25 of title	Q-D
24	BM25 of URL	Q-D
25	BM25 of whole document	Q-D
26	LMIR.ABS of body	Q-D
27	LMIR.ABS of anchor	Q-D
28	LMIR.ABS of title	Q-D
29	LMIR.ABS of URL	Q-D
30	LMIR.ABS of whole document	Q-D
31	LMIR.DIR of body	Q-D
32	LMIR.DIR of anchor	Q-D
33	LMIR.DIR of title	Q-D
34	LMIR.DIR of URL	Q-D
35	LMIR.DIR of whole document	Q-D
36	LMIR.JM of body	Q-D
37	LMIR.JM of anchor	Q-D
38	LMIR.JM of title	Q-D
39	LMIR.JM of URL	Q-D
40	LMIR.JM of whole document	Q-D

49	HITS authority	Q-D
50	HITS hub	Q-D
51	PageRank	D

Web IR - MOSIG

31

# Integration of elements for Web retrieval

- Example: Learning to rank using logistic regression (Pointwise)
- Idea : to learn the best combination of coefficients using a set of values to estimate relevance

- Example (2 params (ex. <PageRank, BM25> as <x1, x2>, 3 coeffs for the learned function) :

$$\text{output} \leftarrow b_0 + b_1 * x_1 + b_2 * x_2$$

- Logistic regression :  $p(\text{class}=1) = 1 / (1 + e^{-\text{output}})$

- Iteration (e.g. using stochastic Gradient Descent)

1. Output using current values of coefficients
2. Re-estimate coeffs based on the prediction error (i items with class  $Y_i$ , j coeffs, m learning rate parameter)
  - $b_j \leftarrow b_j - m * \text{SUM}_i (\text{output}(\text{item}_j) - Y_i) * \text{item}_j(i)$

Web IR - MOSIG

32

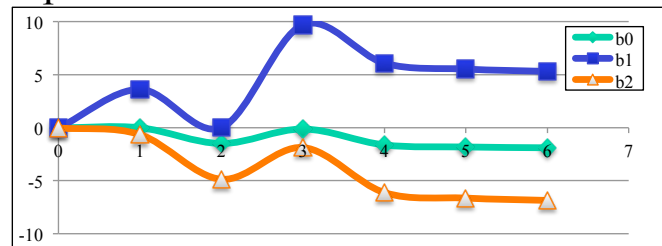


# Integration of elements for Web retrieval

Input (known : Y == relevant)

Item id	x1	x2	Y
0	2,7810836	2,550537003	0
1	1,465489372	2,362125076	0
2	3,396561688	4,400293529	0
3	1,38807019	1,850220317	0
4	3,06407232	3,005305973	0
5	7,627531214	2,759262235	1
6	5,332441248	2,088626775	1
7	6,922596716	1,77106367	1
8	8,675418651	-0,242068655	1
9	7,673756466	3,508563011	1

Optimization



epoch	b0	b1	b2
6	-1,739138611	4,140452807	-7,266752272

At the end of epoch 6,  
decision:

if  $P(\text{class}=1) > 0.5$  then decision=1  
else  $y=0$

Item Id	Y	P(class = 1)	decision
0	0	0,033912686	0
1	0	0,000787008	0
2	0	0,000387722	0
3	0	0,011136864	0
4	0	0,206587548	0
5	1	1	1
6	1	0,999999986	1
7	1	1	1
8	1	1	1
9	1	0,999999972	1

Web IR - MOSIG

# Integration of elements for Web Retrieval

- Others factors

- Correlation study in [http://www.searchmetrics.com/wp-content/uploads/Ranking\\_Correlations\\_EN\\_print.pdf](http://www.searchmetrics.com/wp-content/uploads/Ranking_Correlations_EN_print.pdf)
- Guesses on Google <http://backlinko.com/google-ranking-factors>
- User data (clicks, ...)
- Site speed, extension
- Pages : number of backlinks, size
- Social : refs in Twitter, Google+ ...

# Conclusion

- Short view of web retrieval techniques
  - Web search specificities
- For meta-search (fusion of runs from several search engines), Learning to Rank can be used
- Research problems
  - Integration of users input (logs) to enhance results
  - Dimensionality (users, web pages, queries, languages, dynamicity)
  - Integration of content and semantics (RDF Resource Description Framework)

# Work to be done

- Understand specificities of web search
- Compute the values on HUB/AUTH + Pagerank on the slides example
- Look at Letor 3.0 and learning to rank papers: understand what are the goals of pairwise and listwise learning to rank

# References

- L. Page and S. Brin and R. Motwani and T. Winograd, **The PageRank Citation Ranking: Bringing Order to the Web**, Technical Report. Stanford InfoLab, 1999, <http://infolab.stanford.edu/~backrub/google.html>
- Jon M. Kleinberg, **Authoritative sources in a hyperlinked environment** J. ACM 46, 5, September 1999, <http://www.cs.cornell.edu/home/kleinber/auth.pdf>
- T.-Y. Liu, **Learning to Rank for Information Retrieval**, Foundations and Trends in Information Retrieval Vol. 3, No. 3 (2009): [http://didawiki.di.unipi.it/lib/exe/fetch.php/magistrainformatica/ir/ir13/1\\_-\\_learning\\_to\\_rank.pdf](http://didawiki.di.unipi.it/lib/exe/fetch.php/magistrainformatica/ir/ir13/1_-_learning_to_rank.pdf) (part 1)
- T.-Y. Liu, J. Xu, T. Qin, W.-Y. Xiong, and H. Li, **LETOR: Benchmark dataset for research on learning to rank for information retrieval**, in SIGIR'07 Workshop on Learning to Rank for Information Retrieval, 2007, <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/08/letor3.pdf>