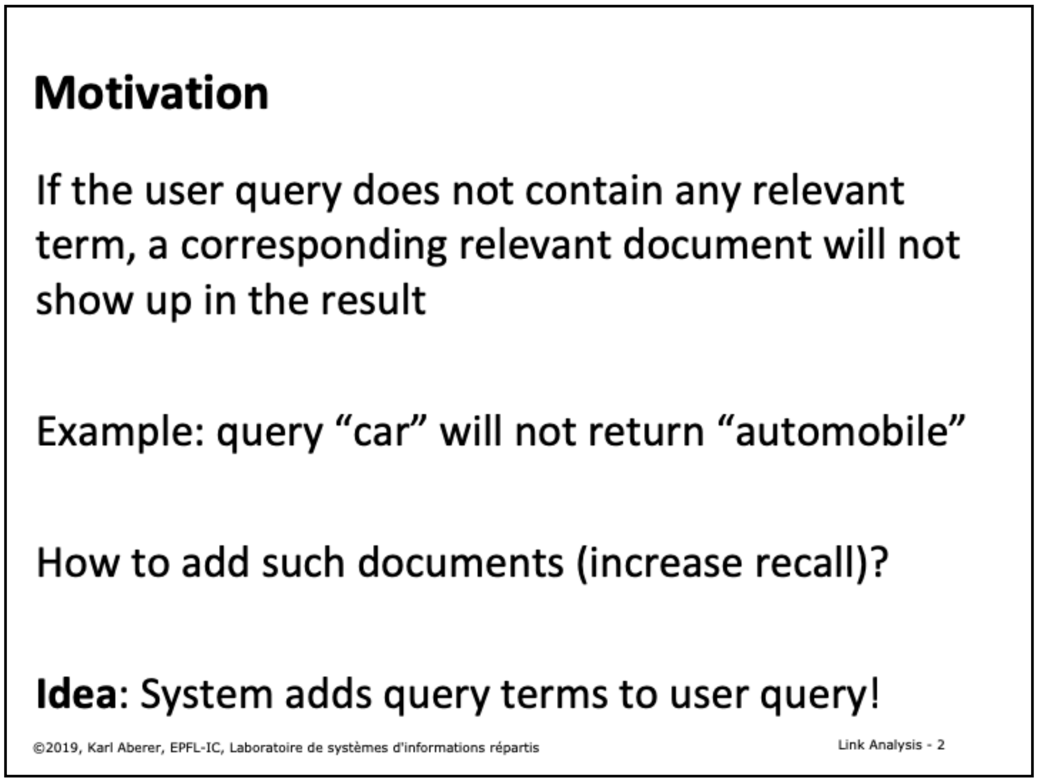
9. QUERY EXPANSION



Motivation

If the user query does not contain any relevant term, a corresponding relevant document will not show up in the result

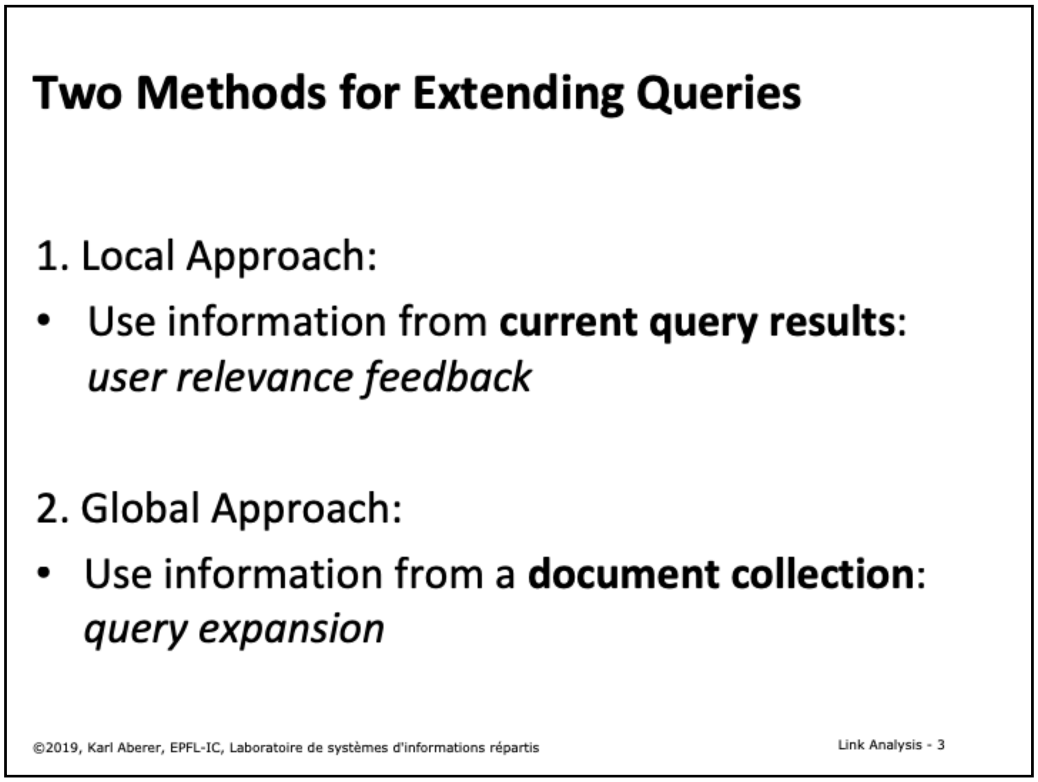
Example: query “car” will not return “automobile”

How to add such documents (increase recall)?

Idea: System adds query terms to user query!

Link Analysis - 2

Users cannot predict or imagine all possible ways of how the concepts they are interested to find in their search can be expressed in natural language. This may have as a consequence, even under the vector space retrieval model, that relevant results are missed. This is, for the example, the case when there exist different synonyms (different terms with the same meaning). In the following we will see one possible approach to deal with this problem, namely extending the user query automatically by the system with additional terms.

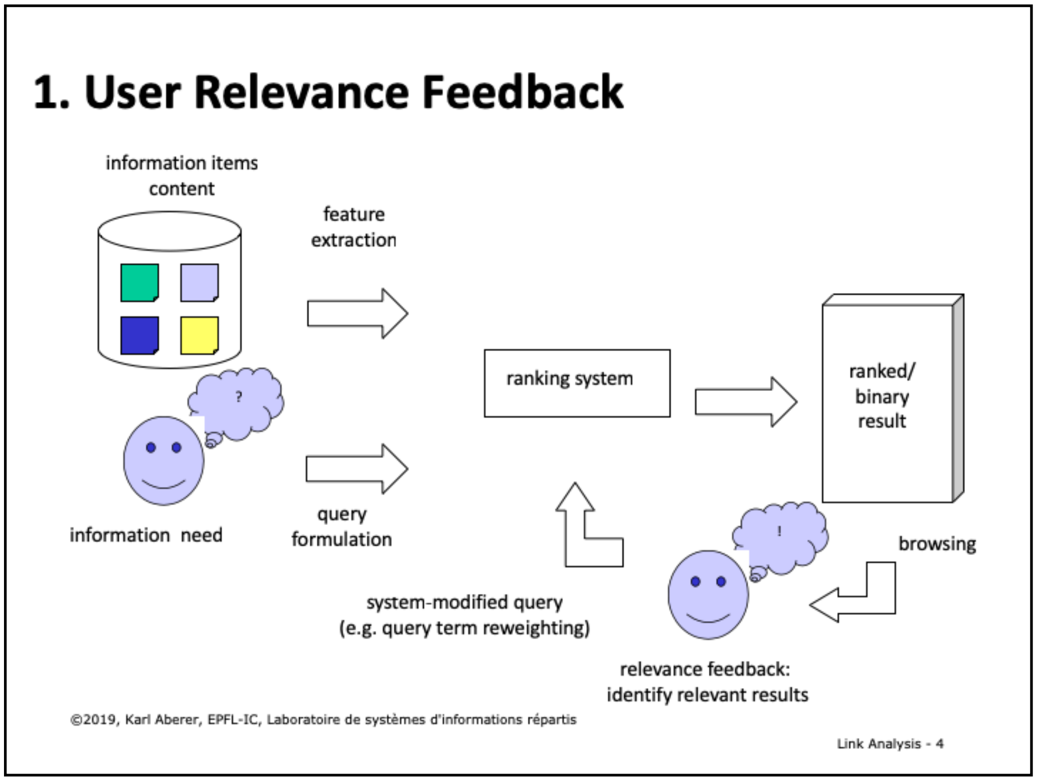


Two Methods for Extending Queries

1. Local Approach:
- Use information from **current query results**: *user relevance feedback*

2. Global Approach:
- Use information from a **document collection**: *query expansion*

In the following we will present two types of approaches to query extension, which are distinguished by the source of information used to identify new additional query terms. In the local approach the source of information is the current user query, respectively results produced by answering the user query. In the global approach the source of information is a existing document collection, either the documents that make up the corpus that is being queried by the user, or another, external collection of documents.



1. User Relevance Feedback

information items content
feature extraction
ranking system
ranked/ binary result
?
information need
query formulation
!
browsing
system-modified query (e.g. query term reweighting)
relevance feedback: identify relevant results

In general, a user does not necessarily know what is his information need and how to appropriately formulate a query. BUT usually a user can well identify relevant documents. Therefore the idea of user relevance feedback is to reformulate a query by taking into account feedback of the user on the relevance of already retrieved documents.

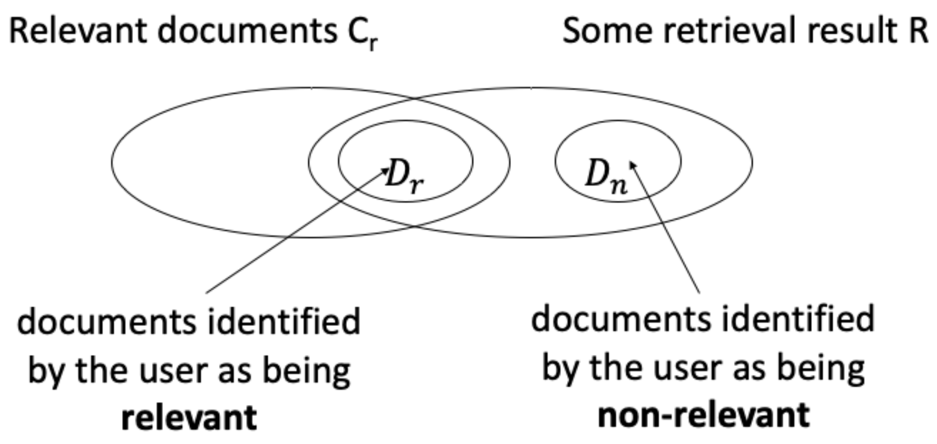
The advantages of such an approach are the following:

•The user is not involved in query formulation, but just points to interesting data items.

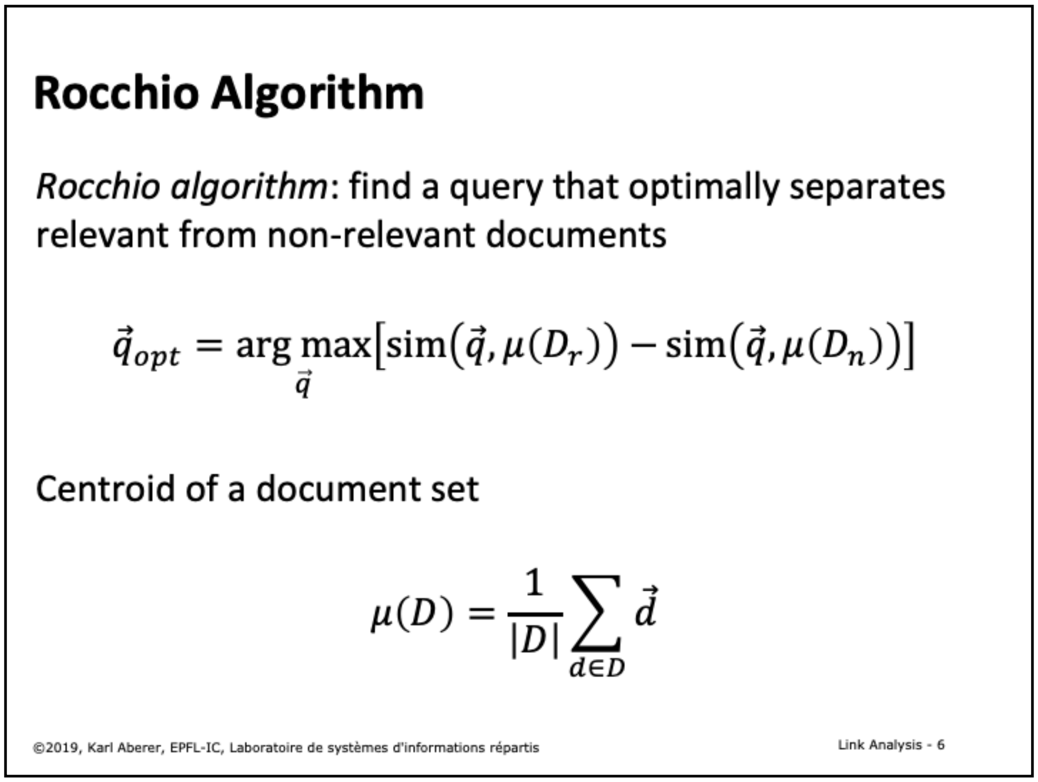
•The search task can be split up in smaller steps.

•The search task becomes a process converging to the desired result.

Feedback from Users


Relevant documents C_r
Some retrieval result R
D_r
D_n
documents identified by the user as being **relevant**
documents identified by the user as being **non-relevant**

The general situation when receiving feedback from users can be depicted as follows: the retrieval system returns some result set R that is presented to the user. This result set overlaps with the set of relevant documents (C_r). The user can the identify within the result set both documents that are relevant and non-relevant. This gives the two feedback sets D_r and D_n.



Rocchio Algorithm

Rocchio algorithm: find a query that optimally separates relevant from non-relevant documents

$$\vec{q}_{opt} = \arg\max_{\vec{q}} \left[\mathrm{sim}\left(\vec{q}, \mu(D_r)\right) - \mathrm{sim}\left(\vec{q}, \mu(D_n)\right)\right]$$

Centroid of a document set

$$\mu(D) = \frac{1}{|D|} \sum_{d \in D} \vec{d}$$

The basic idea for user relevance feedback was introduced by Rocchio. It is based on the observation, that the centroid of all document vectors of a document set D can be considered as the most characteristic representation of the document set. Then one could attempt to construct a query q_{opt} that optimally separates relevant from non-relevant documents. In order to achieve this, the query to be constructed has to have maximal similarity with the set of relevant documents, respectively its centroid, and maximal dissimilarity with the set of non-relevant documents, respectively its centroid. This can be achieved by finding a query that maximizes the difference among these two similarity values.

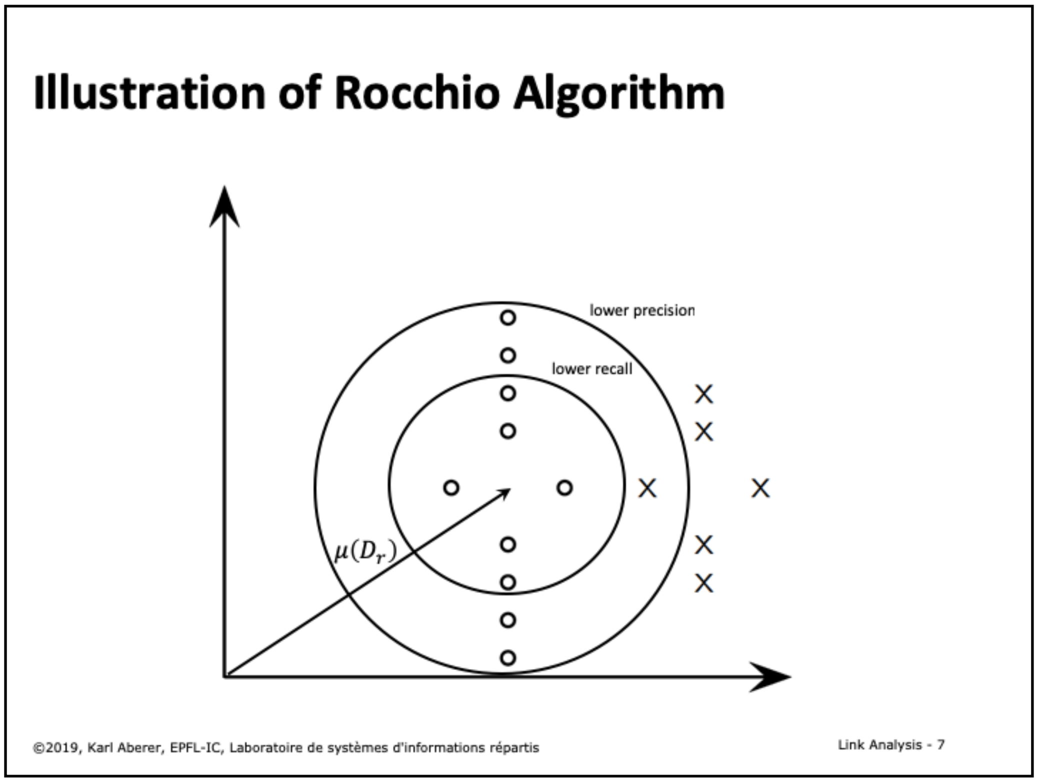


Illustration of Rocchio Algorithm

We now motivate of how the optimal query vector can be found with an illustration. Assume that the relevant documents are marked by circles, and the non-relevant documents are marked by crosses, and that the vector space has (only) 2 dimensions. When we consider the centroid of the relevant documents (which could be a potential query based on user relevance feedback) as a potential search query, then we see that we cannot achieve optimal precision and recall at the same time.

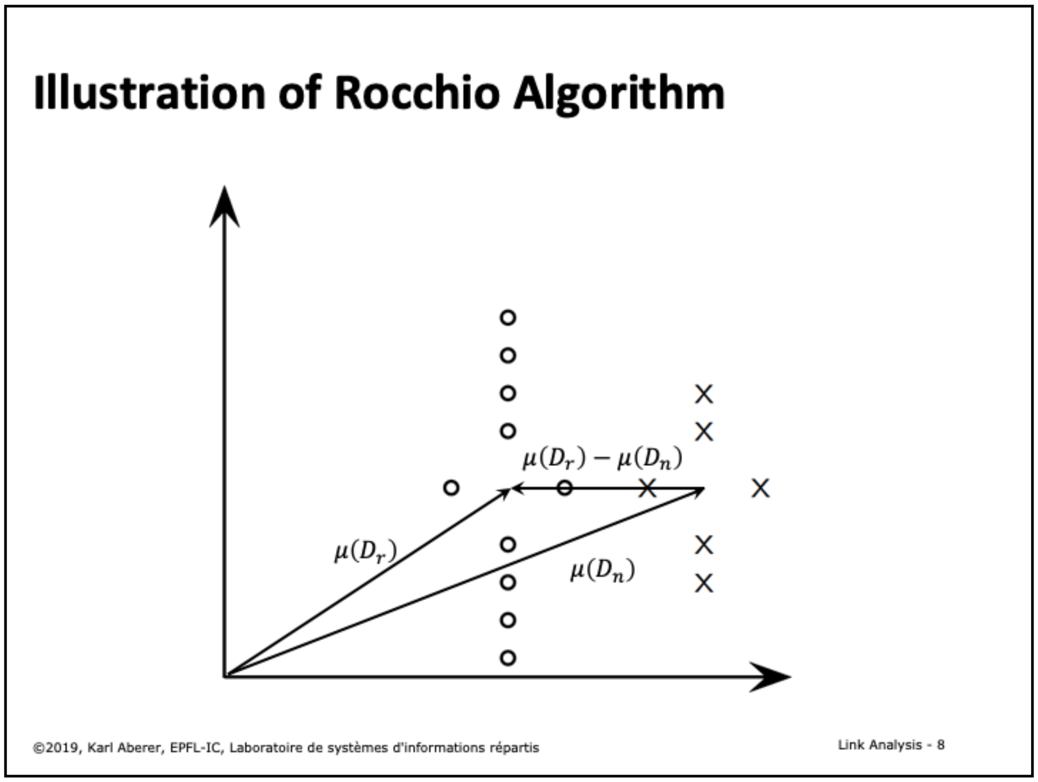
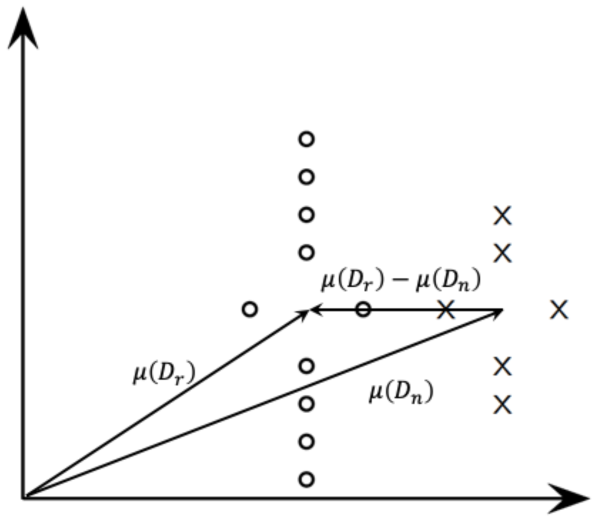


Illustration of Rocchio Algorithm



©2019, Karl Aberer, EPFL-IC, Laboratoire de systèmes d'informations répartis

Link Analysis - 8

We therefore consider also the centroid of the non-relevant documents as part of the user relevance feedback. We compute the difference vector among the two centroids, and we will use this difference vector to “move away” the query from the non-relevant documents.

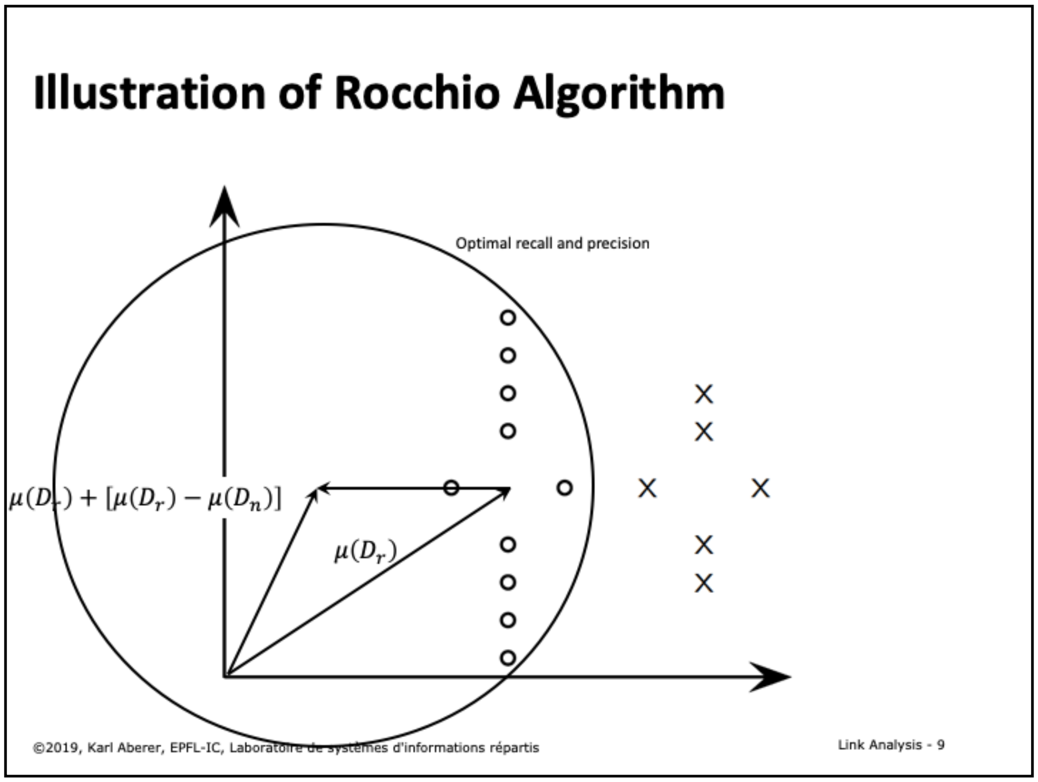
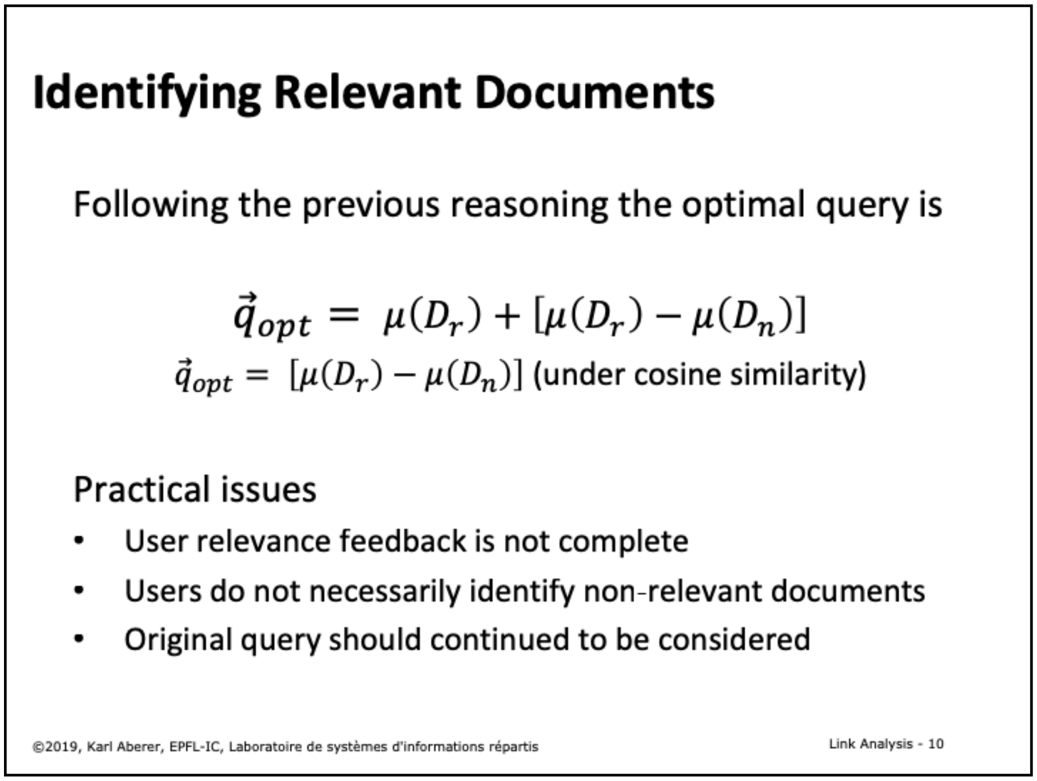


Illustration of Rocchio Algorithm

Optimal recall and precision
$\mu(D_r) + [\mu(D_r) - \mu(D_n)]$
$\mu(D_r)$

We add the difference vector to the centroid for the relevant documents. The resulting optimal query vector now can include all relevant documents in its result, without including non-relevant ones. In practice, such a clear separation will not always be possible, but it has been shown that under some additional assumptions, this method is the optimal way to constructing the optimal query vector.



Identifying Relevant Documents

Following the previous reasoning the optimal query is

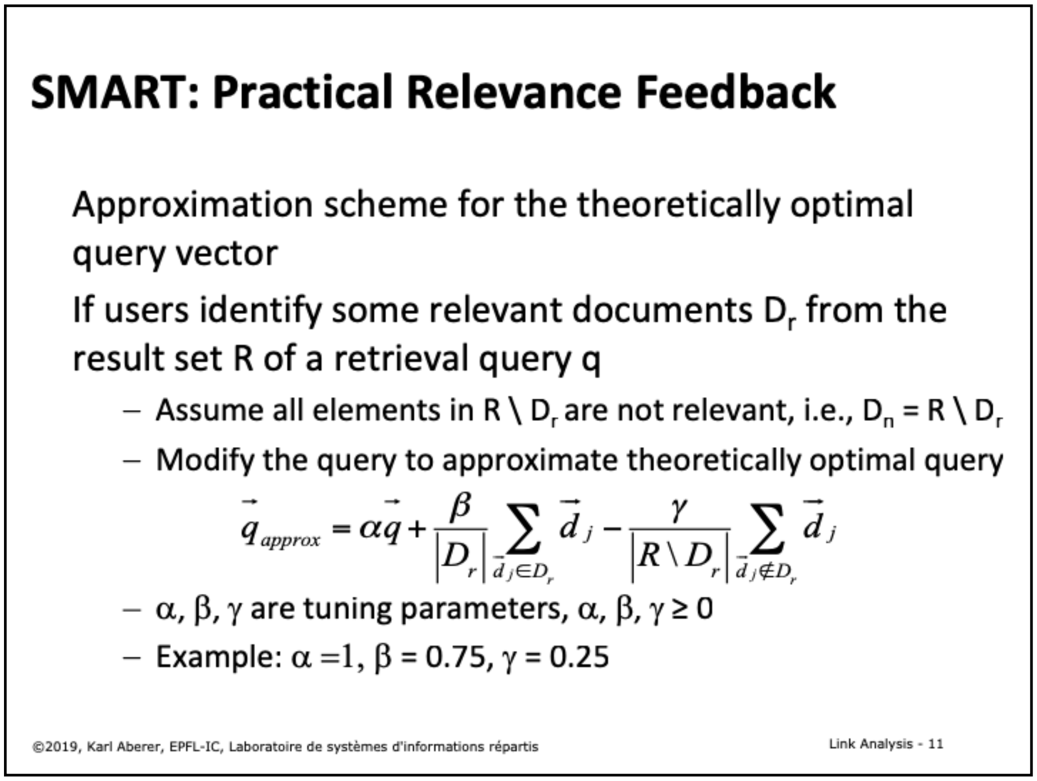
$$\vec{q}_{opt} = \mu(D_r) + [\mu(D_r) - \mu(D_n)]$$

$$\vec{q}_{opt} = [\mu(D_r) - \mu(D_n)] \text{ (under cosine similarity)}$$

Practical issues

- User relevance feedback is not complete
- Users do not necessarily identify non-relevant documents
- Original query should continued to be considered

Constructing an optimal query vector as described before is only theoretically possible, since the complete information on relevant and non-relevant documents is lacking in practice. Therefore, the theoretical considerations put forward so far, serve as an intuition to devise a practical scheme, that is **approximating** the theoretical construction of an optimal query vector.



SMART: Practical Relevance Feedback

Approximation scheme for the theoretically optimal query vector

If users identify some relevant documents D_r from the result set R of a retrieval query q

- Assume all elements in $R \setminus D_r$ are not relevant, i.e., $D_n = R \setminus D_r$
- Modify the query to approximate theoretically optimal query

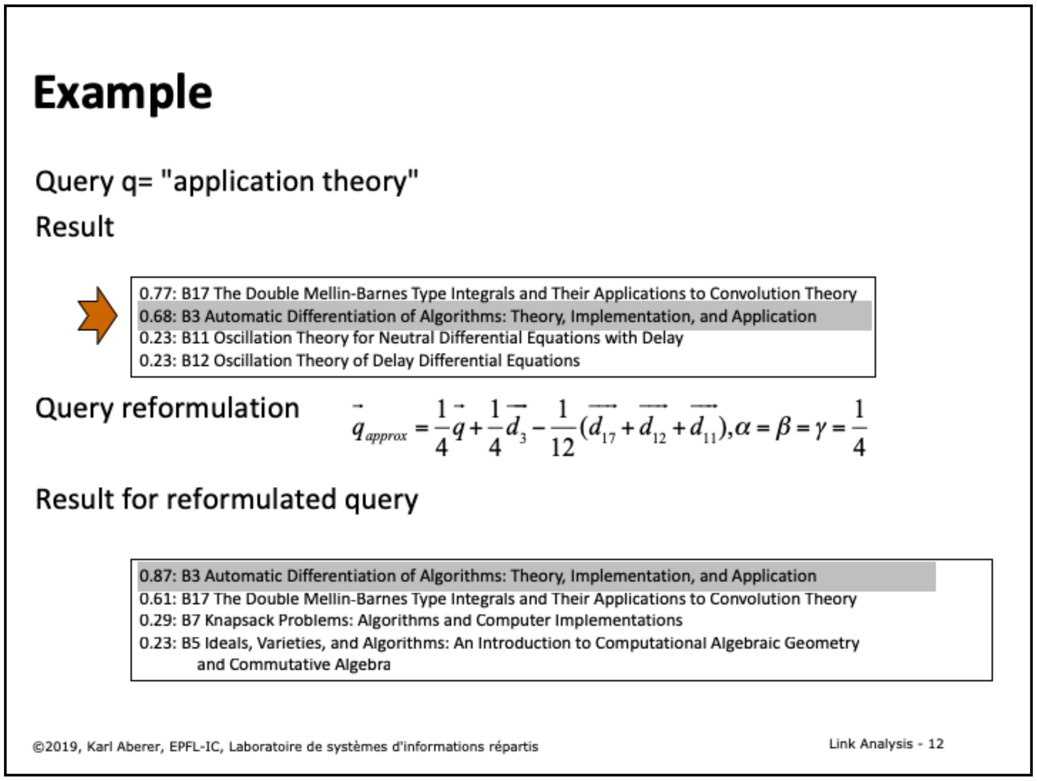
$$\vec{q}_{approx} = \alpha\vec{q} + \frac{\beta}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \frac{\gamma}{|R \setminus D_r|}\sum_{\vec{d}_j \notin D_r}\vec{d}_j$$

- α, β, γ are tuning parameters, $\alpha, \beta, \gamma \geq 0$
- Example: $\alpha = 1$, $\beta = 0.75$, $\gamma = 0.25$

The approximation scheme for user relevance feedback is called SMART. It starts from the assumption that users have identified some relevant documents. Then the scheme assumes that all other documents should be considered as non-relevant. This results in a modification of the original query that is controlled by 3 tuning parameters.

Since this is of course not correct, two mechanisms are used to moderate the impact of this wrong assumption:

1. The original query vector is maintained, in order not to drift away too dramatically from the original user query.
2. The weight given for the modification using the centroid of non-relevant documents is generally kept lower than the weight for the centroid of the relevant documents, as their non-relevance is just an assumption made, and not based on real user relevance feedback.



Example

Query q= "application theory"

Result

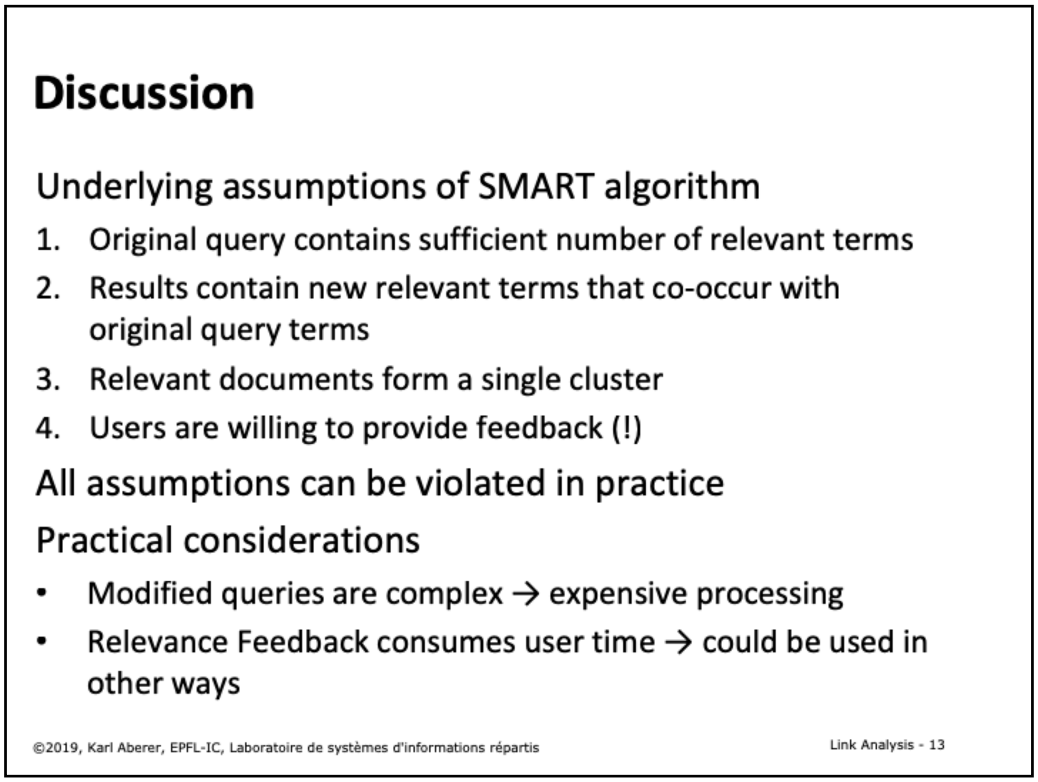
0.77: B17 The Double Mellin-Barnes Type Integrals and Their Applications to Convolution Theory
0.68: B3 Automatic Differentiation of Algorithms: Theory, Implementation, and Application
0.23: B11 Oscillation Theory for Neutral Differential Equations with Delay
0.23: B12 Oscillation Theory of Delay Differential Equations

Query reformulation $\vec{q}_{approx} = \frac{1}{4}\vec{q} + \frac{1}{4}\vec{d_3} - \frac{1}{12}(\vec{d_{17}} + \vec{d_{12}} + \vec{d_{11}}), \alpha = \beta = \gamma = \frac{1}{4}$

Result for reformulated query

0.87: B3 Automatic Differentiation of Algorithms: Theory, Implementation, and Application
0.61: B17 The Double Mellin-Barnes Type Integrals and Their Applications to Convolution Theory
0.29: B7 Knapsack Problems: Algorithms and Computer Implementations
0.23: B5 Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra

This example shows how the query reformulation works. By identifying document B3 as being relevant and modifying the query vector it turns out that new documents (B5 and B7) become relevant. The reason is that those new documents share terms with document B3, and these terms are newly considered in the reformulated query.


Discussion

Underlying assumptions of SMART algorithm

1. Original query contains sufficient number of relevant terms
2. Results contain new relevant terms that co-occur with original query terms
3. Relevant documents form a single cluster
4. Users are willing to provide feedback (!)

All assumptions can be violated in practice

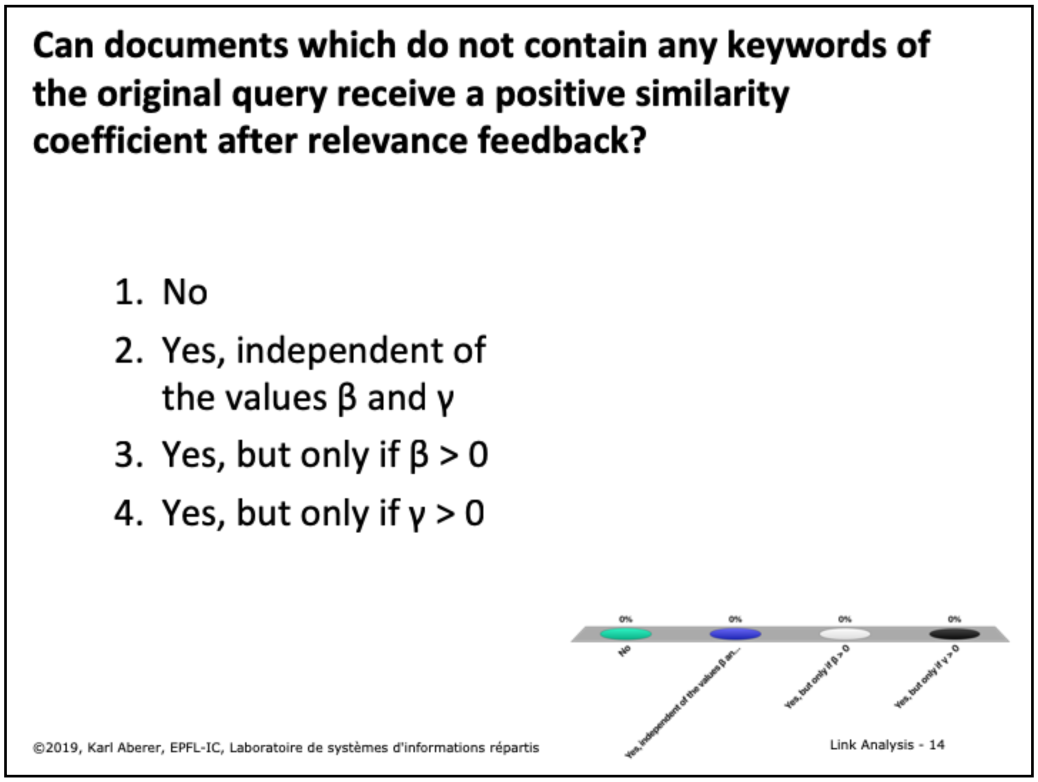
Practical considerations

- Modified queries are complex → expensive processing
- Relevance Feedback consumes user time → could be used in other ways

Concerning the first assumption, if the initial query of the user does not contain sufficient information to retrieve at least a few documents that are relevant to the true interest of the user, the relevance feedback system will not be able to produce sufficient relevant documents with additional terms.

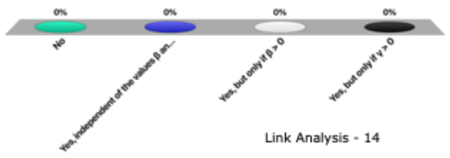
Concerning the second assumption, new terms can only be included as part of the modified query, if they co-occur at least in some documents together with original query terms. Otherwise, these terms could never be part of relevant documents in the result of the original query (why?).

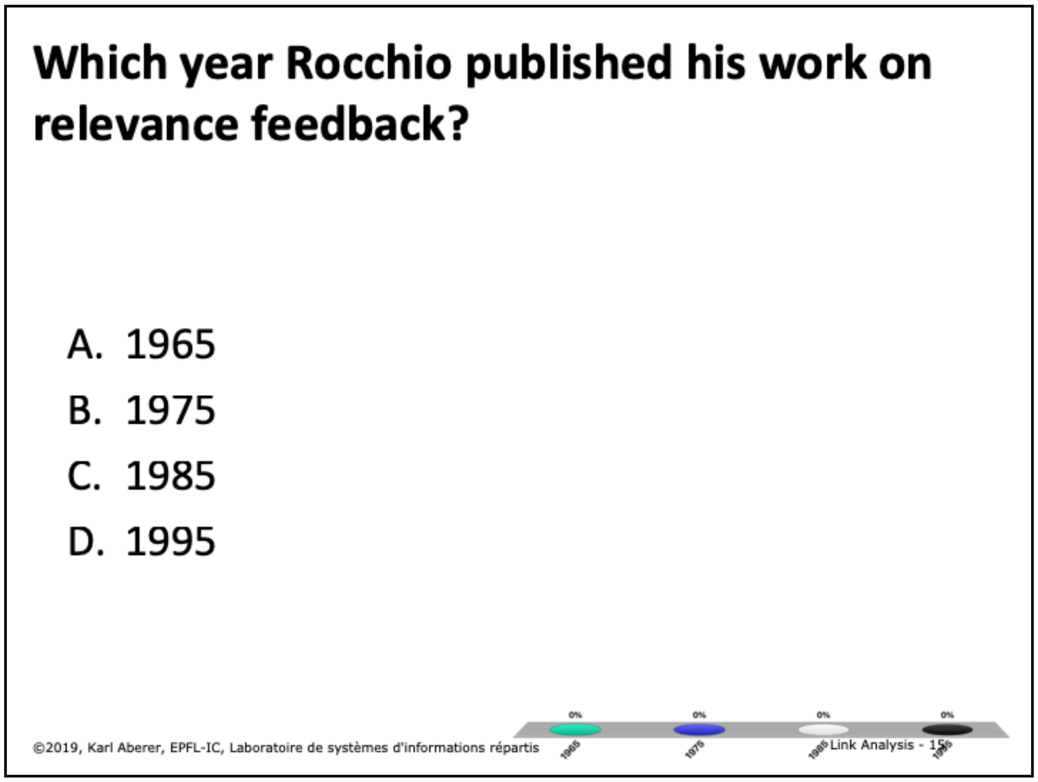
Concerning the third assumption, implicitly the SMART algorithm assumes that all relevant documents are part of one cluster in the vector space. If they form multiple clusters, it is not able to correctly produce a query that can retrieve the relevant documents.



Can documents which do not contain any keywords of the original query receive a positive similarity coefficient after relevance feedback?

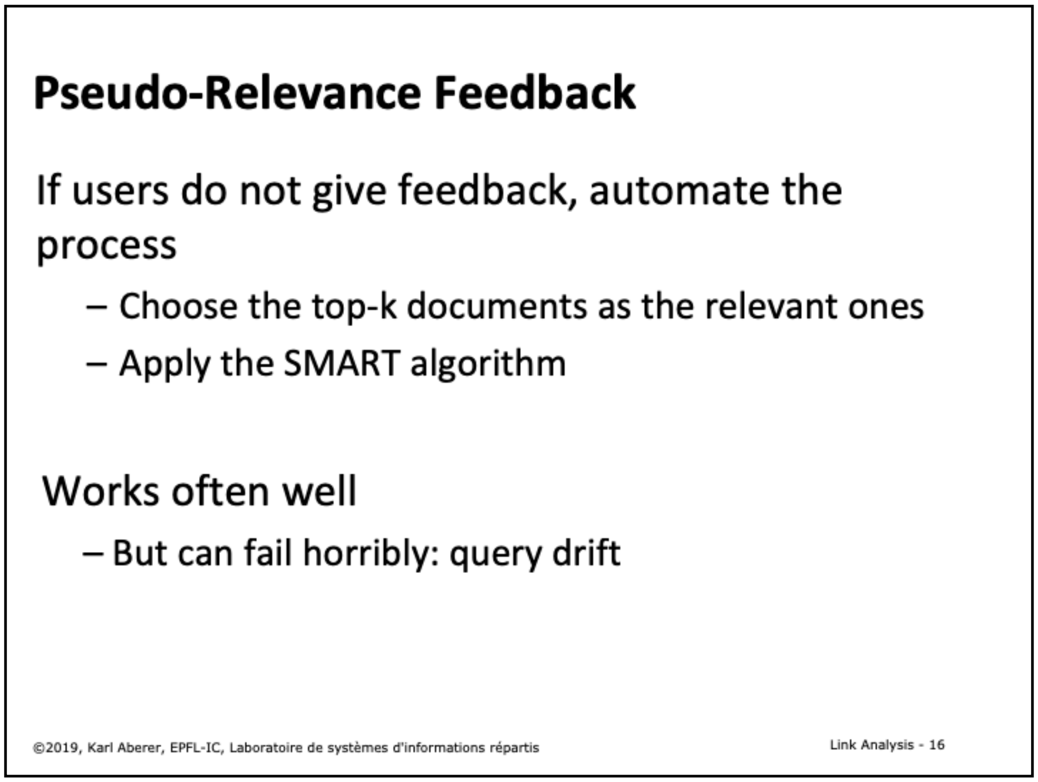
1. No
2. Yes, independent of the values β and γ
3. Yes, but only if β > 0
4. Yes, but only if γ > 0





Which year Rocchio published his work on relevance feedback?

A. 1965
B. 1975
C. 1985
D. 1995



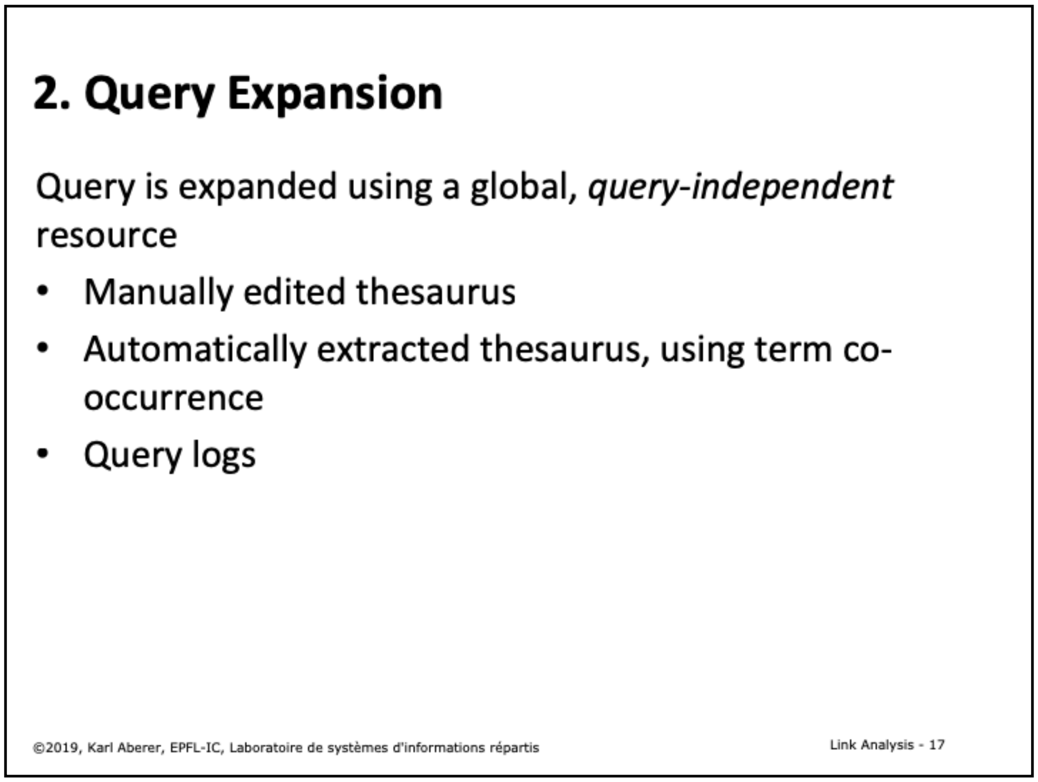
Pseudo-Relevance Feedback

If users do not give feedback, automate the process

- Choose the top-k documents as the relevant ones
- Apply the SMART algorithm

Works often well

- But can fail horribly: query drift

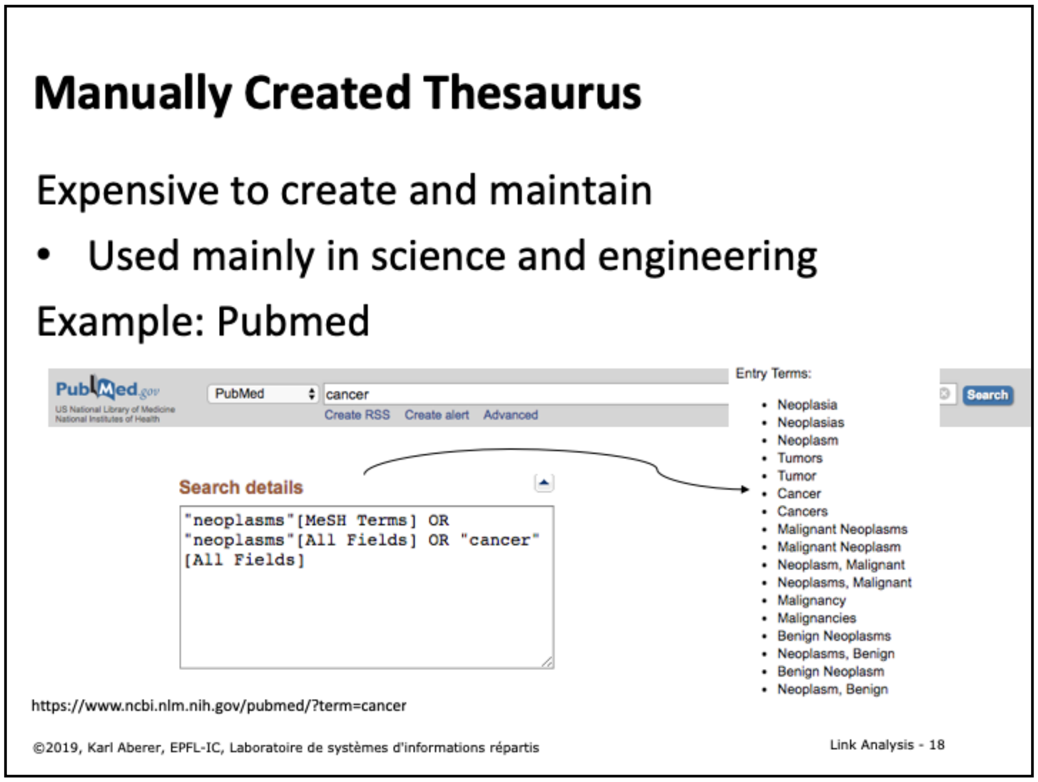


2. Query Expansion

Query is expanded using a global, *query-independent* resource

- Manually edited thesaurus
- Automatically extracted thesaurus, using term co-occurrence
- Query logs

Global methods for expanding user queries can rely on a variety of resources. These may include thesauri (a thesaurus is a database that contains (near-) synonyms) that are manually constructed or automatically derived, or the automated analysis of query logs.



Manually Created Thesaurus

Expensive to create and maintain

- Used mainly in science and engineering

Example: Pubmed

PubMed.gov
US National Library of Medicine
National Institutes of Health
PubMed
cancer
Create RSS
Create alert
Advanced
Search

Search details

```
"neoplasms"[MeSH Terms] OR
"neoplasms"[All Fields] OR "cancer"
[All Fields]
```

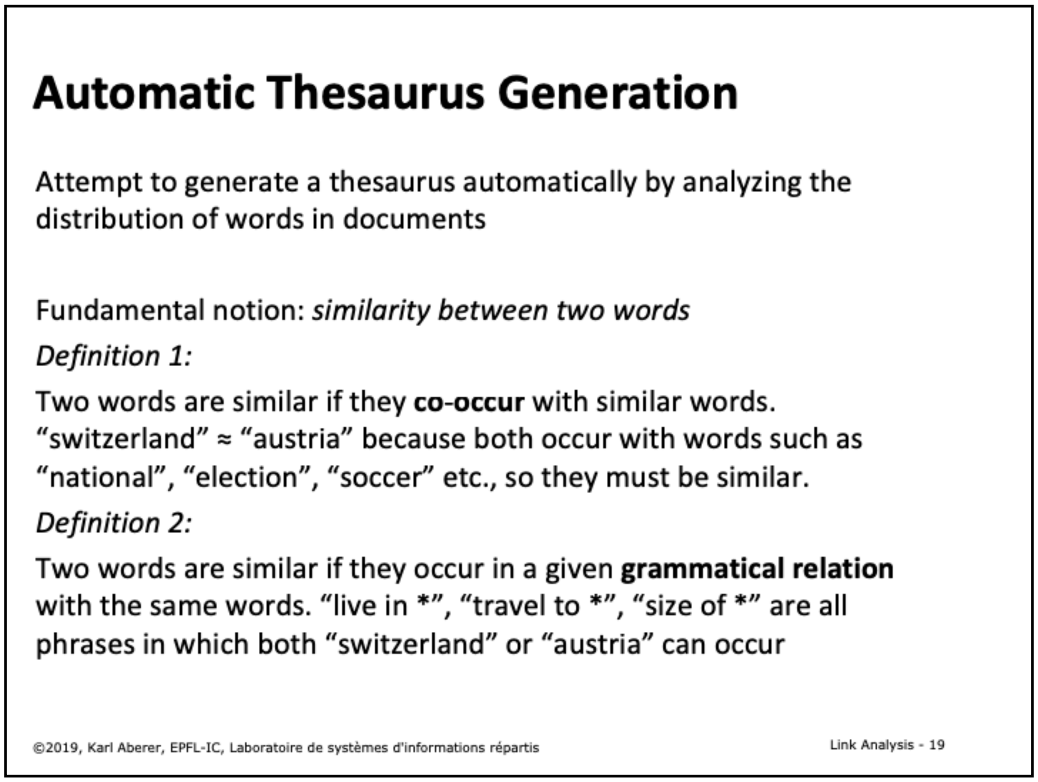
Entry Terms:

- Neoplasia
- Neoplasias
- Neoplasm
- Tumors
- Tumor
- Cancer
- Cancers
- Malignant Neoplasms
- Malignant Neoplasm
- Neoplasm, Malignant
- Neoplasms, Malignant
- Malignancy
- Malignancies
- Benign Neoplasms
- Neoplasms, Benign
- Benign Neoplasm
- Neoplasm, Benign

https://www.ncbi.nlm.nih.gov/pubmed/?term=cancer

Link Analysis - 18

Performing query expansion using a manually thesaurus requires the (expensive) effort of creating and maintaining such a thesaurus. This task is mainly performed in highly specialized technical fields in science and engineering. One prominent example of such a Thesaurus is maintained by Pubmed, the biggest publication database for medical literature maintained by the NIH, the National Institute of Health in the US. When using its search engine, you will find a window ”Search details” that shows how the user query is automatically expanded using the Pubmed thesaurus. In this example we see that the search system identifies that “cancer” is an entry on the concept “neoplasms”, and thus extends the query with all entries that it finds associated in the thesaurus (e.g. it would also search for “tumor”).



Automatic Thesaurus Generation

Attempt to generate a thesaurus automatically by analyzing the distribution of words in documents

Fundamental notion: *similarity between two words*

Definition 1:

Two words are similar if they **co-occur** with similar words. “switzerland” ≈ “austria” because both occur with words such as “national”, “election”, “soccer” etc., so they must be similar.

Definition 2:

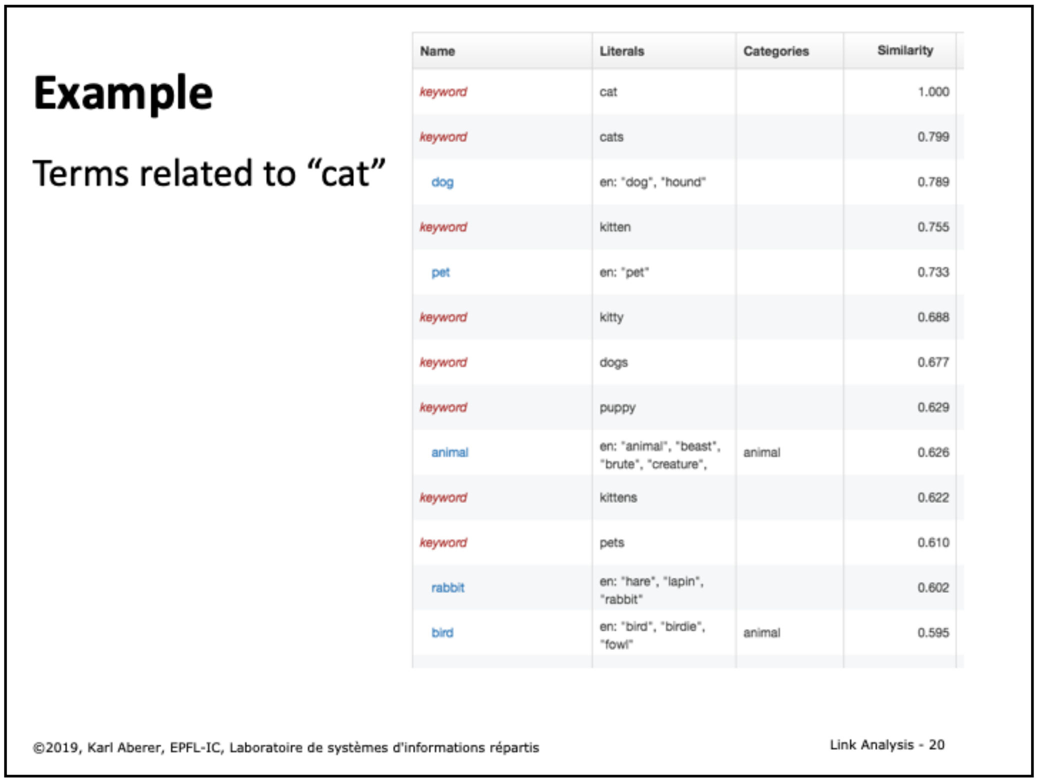
Two words are similar if they occur in a given **grammatical relation** with the same words. “live in *”, “travel to *”, “size of *” are all phrases in which both “switzerland” or “austria” can occur

©2019, Karl Aberer, EPFL-IC, Laboratoire de systèmes d'informations répartis

Link Analysis - 19

In order to avoid the effort of manually creating a thesaurus one can attempt to create it automatically by studying large numbers of documents and the distribution of words in those. This leads to the concept of similarity of words. There exists two basic methods to study this similarity, either purely statistically, by observing which words occur together in documents, or in a more accurate way by identifying whether the words occur in the same grammatical relationships.

We will later in the course study both types of methods. For the first approach we will study so-called “word embeddings”. For the second approach we will learn about methods of “information extraction”.

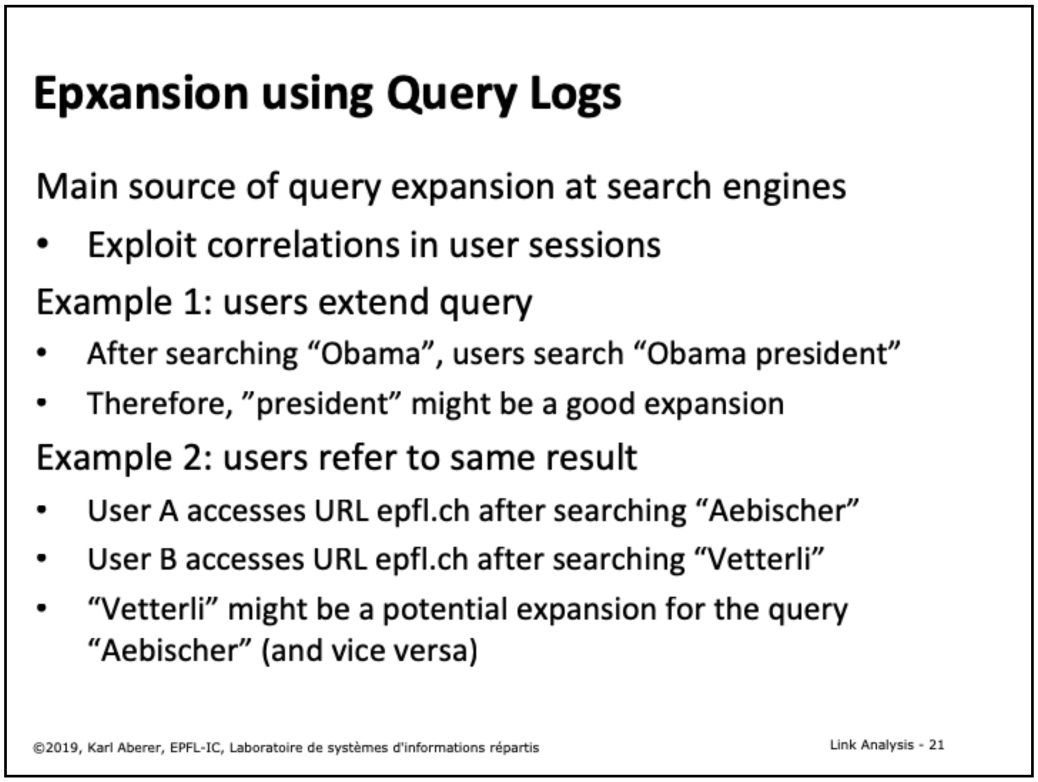


Example

Terms related to “cat”

Name	Literals	Categories	Similarity
keyword	cat		1.000
keyword	cats		0.799
dog	en: "dog", "hound"		0.789
keyword	kitten		0.755
pet	en: "pet"		0.733
keyword	kitty		0.688
keyword	dogs		0.677
keyword	puppy		0.629
animal	en: "animal", "beast", "brute", "creature",	animal	0.626
keyword	kittens		0.622
keyword	pets		0.610
rabbit	en: "hare", "lapin", "rabbit"		0.602
bird	en: "bird", "birdie", "fowl"	animal	0.595

This example illustrates of how such automatic thesaurus generation based on statistical analysis looks in practice. A statistical analysis would allow to compute similarity of words. With such a similarity measure it is possible to search for related words (just like searching for documents with an information retrieval system). As the example shows, such a search reveals immediately many terms directly related with the original word, which was “cat”.



Epxansion using Query Logs

Main source of query expansion at search engines

- Exploit correlations in user sessions

Example 1: users extend query

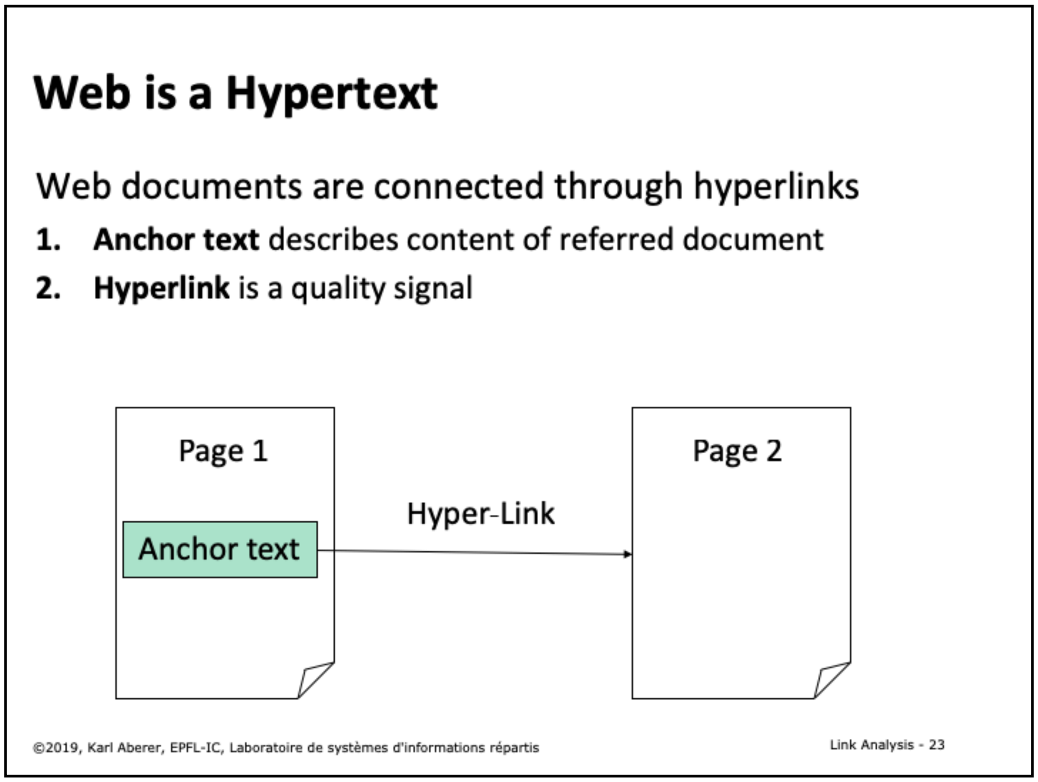
- After searching “Obama”, users search “Obama president”
- Therefore, ”president” might be a good expansion

Example 2: users refer to same result

- User A accesses URL epfl.ch after searching “Aebischer”
- User B accesses URL epfl.ch after searching “Vetterli”
- “Vetterli” might be a potential expansion for the query “Aebischer” (and vice versa)

Query logs contain potentially rich information for query expansion. There a numerous ways of how such knowledge can be exploited. We show here two possible examples. Other methods rely on mining query logs using various techniques, including clustering and association rule mining, that we will encounter in the later part on data mining.

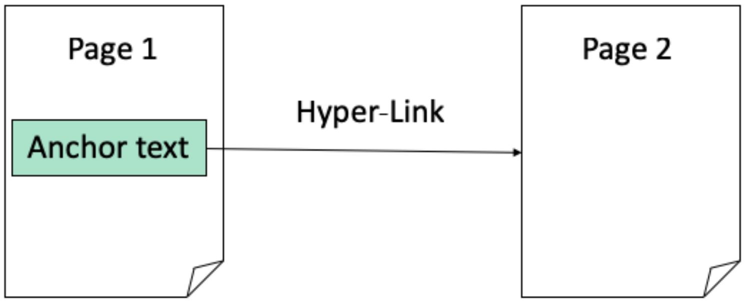
10. LINK-BASED RANKING



Web is a Hypertext

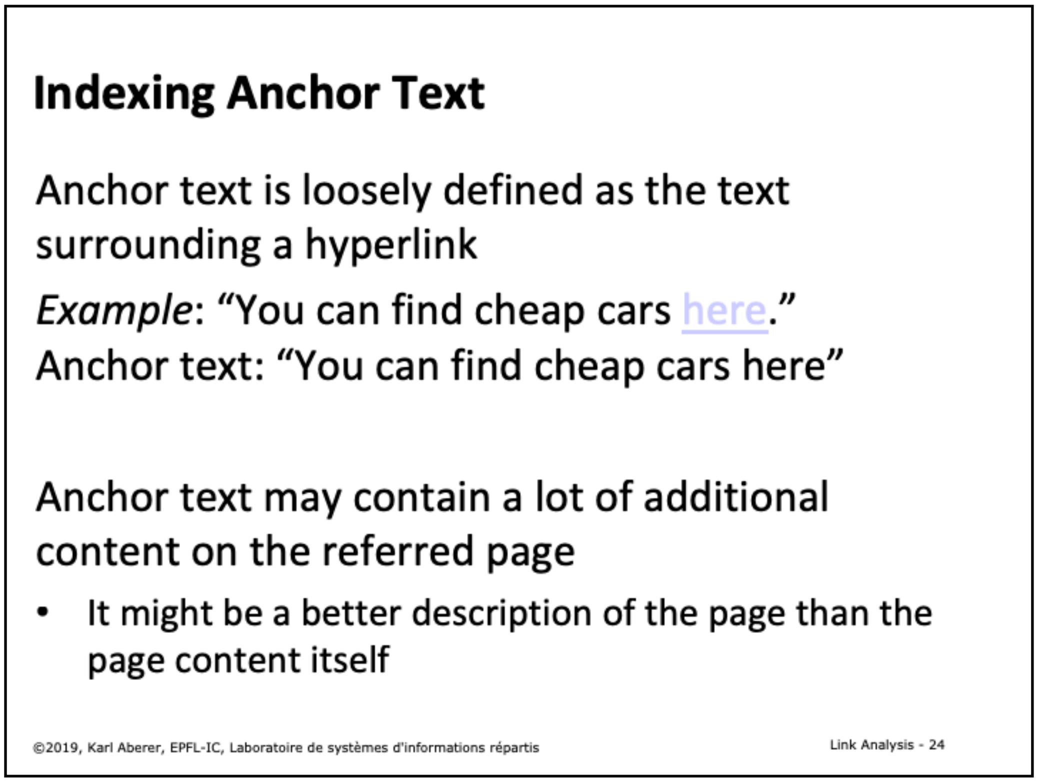
Web documents are connected through hyperlinks

1. **Anchor text** describes content of referred document
2. **Hyperlink** is a quality signal



Beyond the textual contents, Web documents consist also of hyperlinks. A hyperlink can be exploited for information retrieval in two ways:

1. The link is surrounded by some textual information that presumably refers to the content of the document the link is pointing to. Thus this text can complement the content of the referred document.
2. The link can also be considered as an endorsement of the referenced document by the author of the referring document. Thus the link can be used as a signal for quality and importance of the referred document.



Indexing Anchor Text

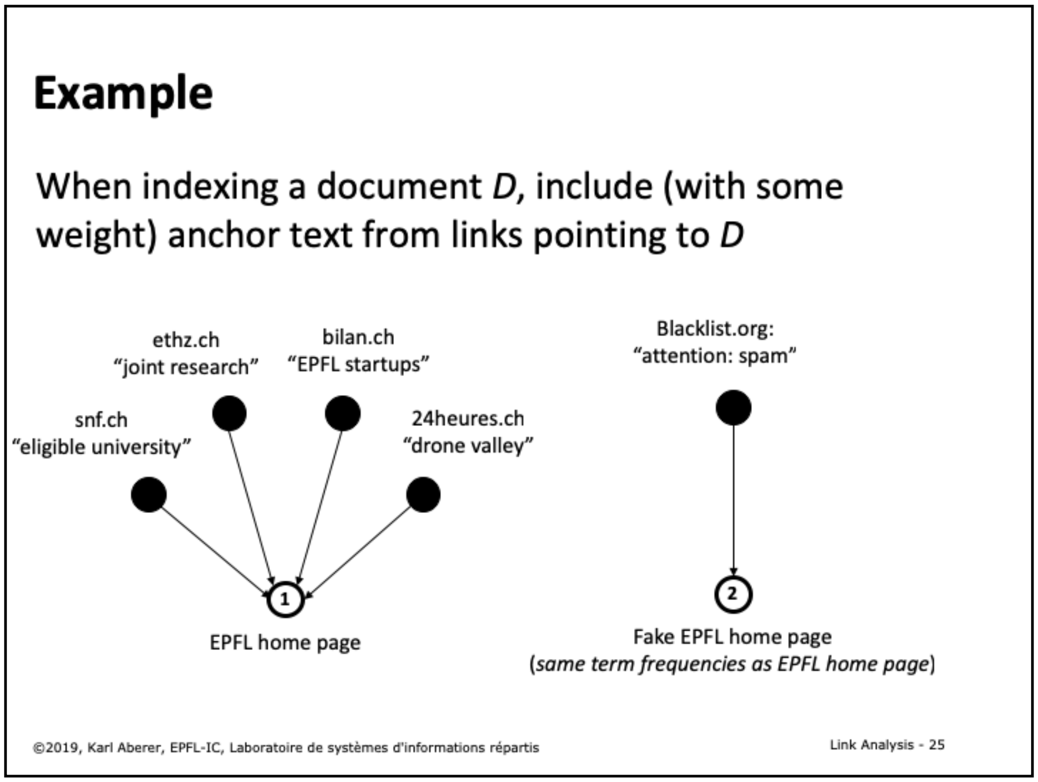
Anchor text is loosely defined as the text surrounding a hyperlink

Example: “You can find cheap cars here.”
Anchor text: “You can find cheap cars here”

Anchor text may contain a lot of additional content on the referred page

- It might be a better description of the page than the page content itself

Anchor text can be defined in different ways. In general, it is considered as the text that is surrounding the link, and not only the text contained as part of the link tag (in the example, this text would simply be “here”.) This text can contain valuable information on the referred page and thus be very helpful in retrieval.



Example

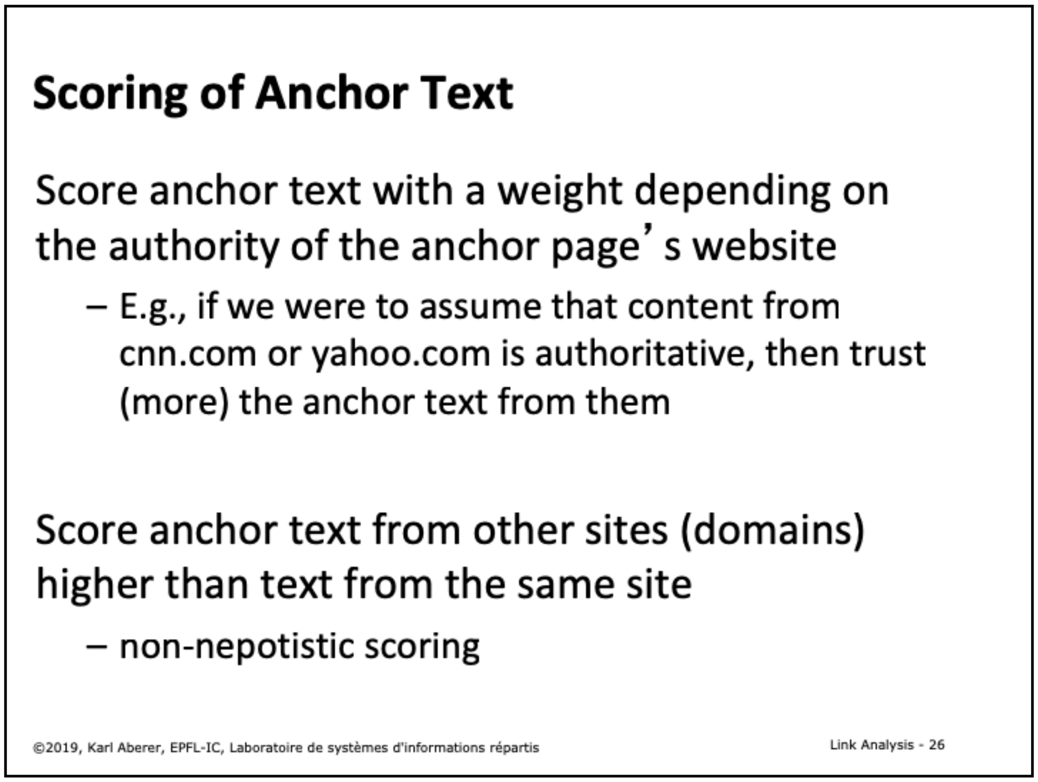
When indexing a document *D*, include (with some weight) anchor text from links pointing to *D*

ethz.ch
"joint research"
bilan.ch
"EPFL startups"
snf.ch
"eligible university"
24heures.ch
"drone valley"
1
EPFL home page
Blacklist.org:
"attention: spam"
2
Fake EPFL home page
(*same term frequencies as EPFL home page*)

This example illustrates the use of anchor text in retrieval. Typically a home page is very graphical and contains often very little relevant content. If we consider a home page, such as the EPFL home page, we would probably found many pages that very well characterize EPFL, such as pages mentioning topics related to research and technology transfer. Typically this links would point to the EPFL home page. Thus it would also let the home page stand out from other EPFL pages (such as pages on the laws and ordonnances of EPFL).

Assume that a malicious Internet user would create a fake EPFL home page. Then chances that such a page is referred by reputed organizations, such as SNF, is very low. On the other hand pages listing spam pages might point to such a page and indicate its true character.

Therefore it makes a lot of sense to include such anchor text in the representation of the document. This is usually done by adding it with a given weight.



Scoring of Anchor Text

Score anchor text with a weight depending on the authority of the anchor page's website

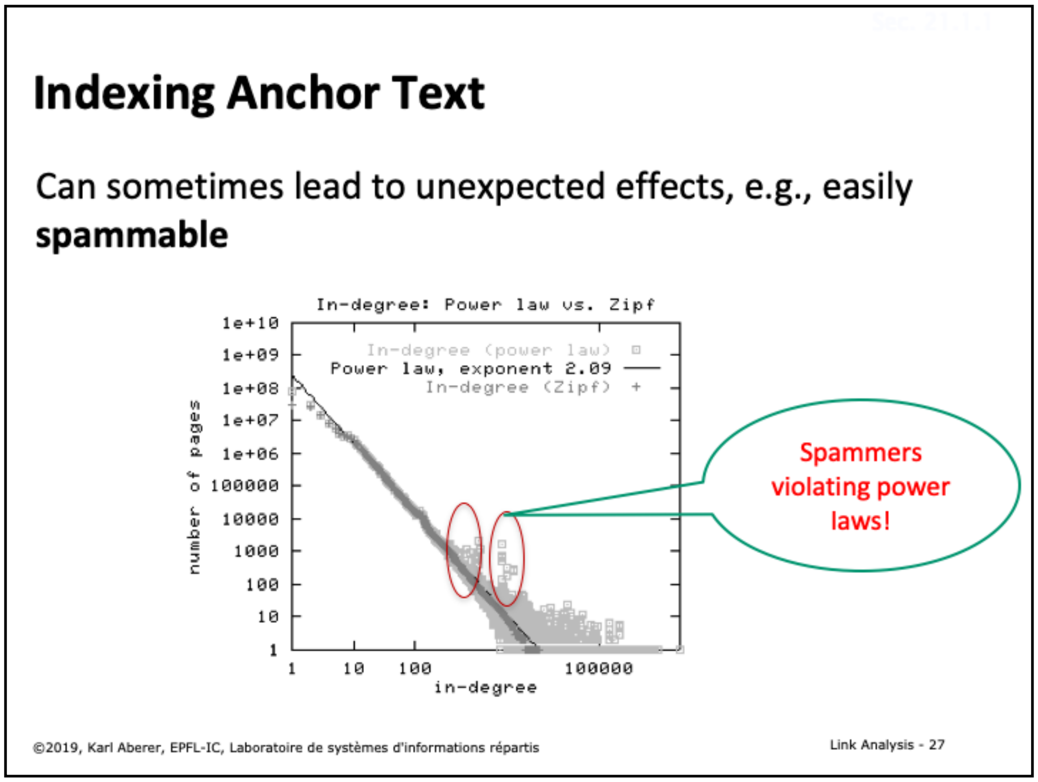
- E.g., if we were to assume that content from cnn.com or yahoo.com is authoritative, then trust (more) the anchor text from them

Score anchor text from other sites (domains) higher than text from the same site

- non-nepotistic scoring

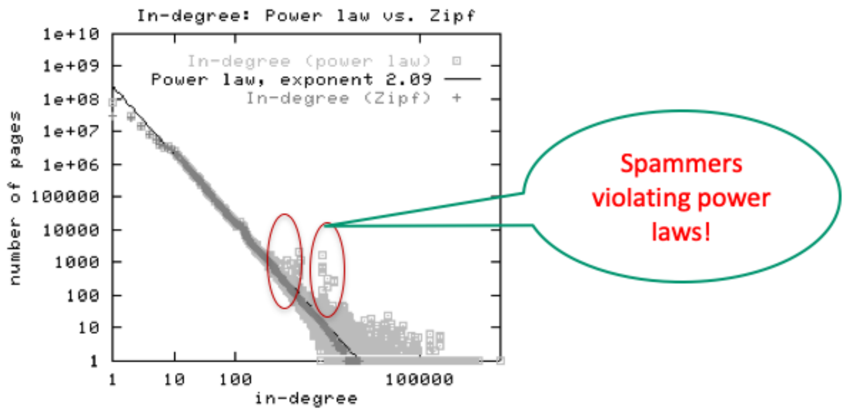
To fight spam, on can adapt the weighting function to the reputation of the page that is referring. This could be done by having a directory of highly reputed sites and to give high weights to its members. A more fundamental of producing such reputation scores we will see in the following part on link-based ranking.

In order to avoid self-promotion, another method to fight link spamming is to give lower weights to links within the same site (nepotistic scoring = promoting your own family members).



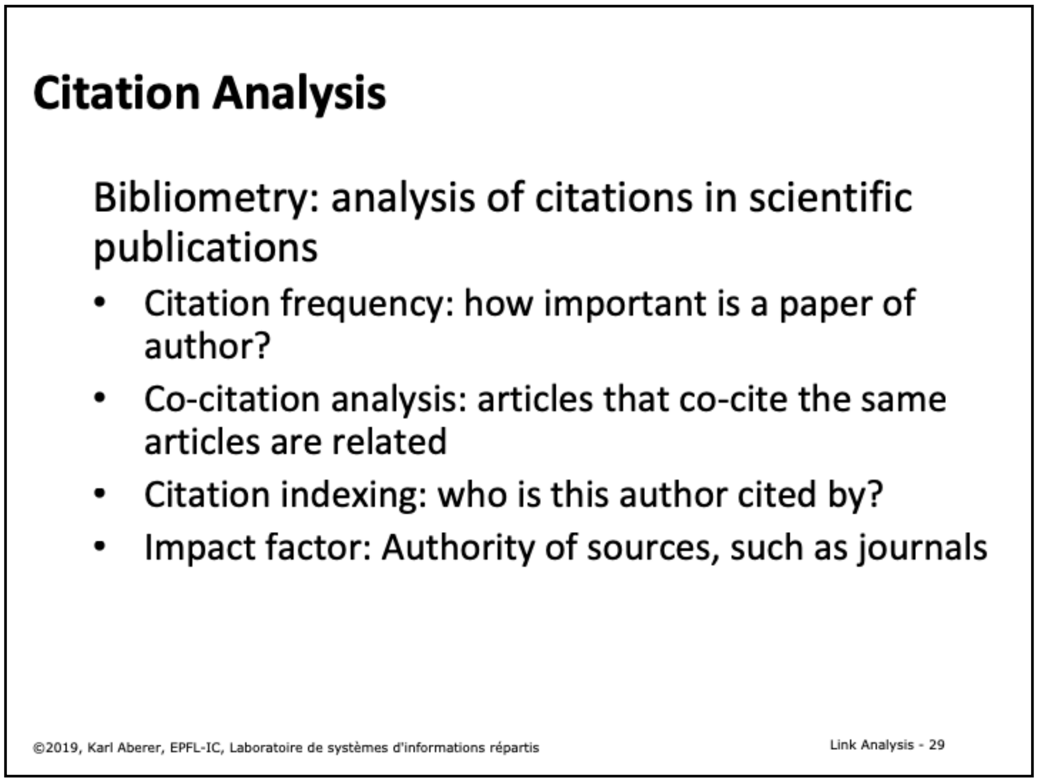
Indexing Anchor Text

Can sometimes lead to unexpected effects, e.g., easily **spammable**



One of the risks of including anchor text is that it makes pages spammable. Malicious users could create spam pages that point to web pages and try to relate it to contents that serve their interests (e.g., higher the quality of preferred pages by adding links, lower the quality of the undesired page by attaching negative anchor text). That this is happening can be seen from analyzing the in-degree distribution of Web pages. The figure shows a standard log-log representation of the in-degree vs. the frequency of pages. Normally this relationship should follow a power-law, which shows in a log-log representation as a linear dependency. In real Web data, we see that this power law is violated, and that certain levels of in-degrees are over-represented. This can be attributed to link spamming, which does create moderate numbers of additional links on Web pages.

LINK-BASED RANKING: PAGERANK



Citation Analysis

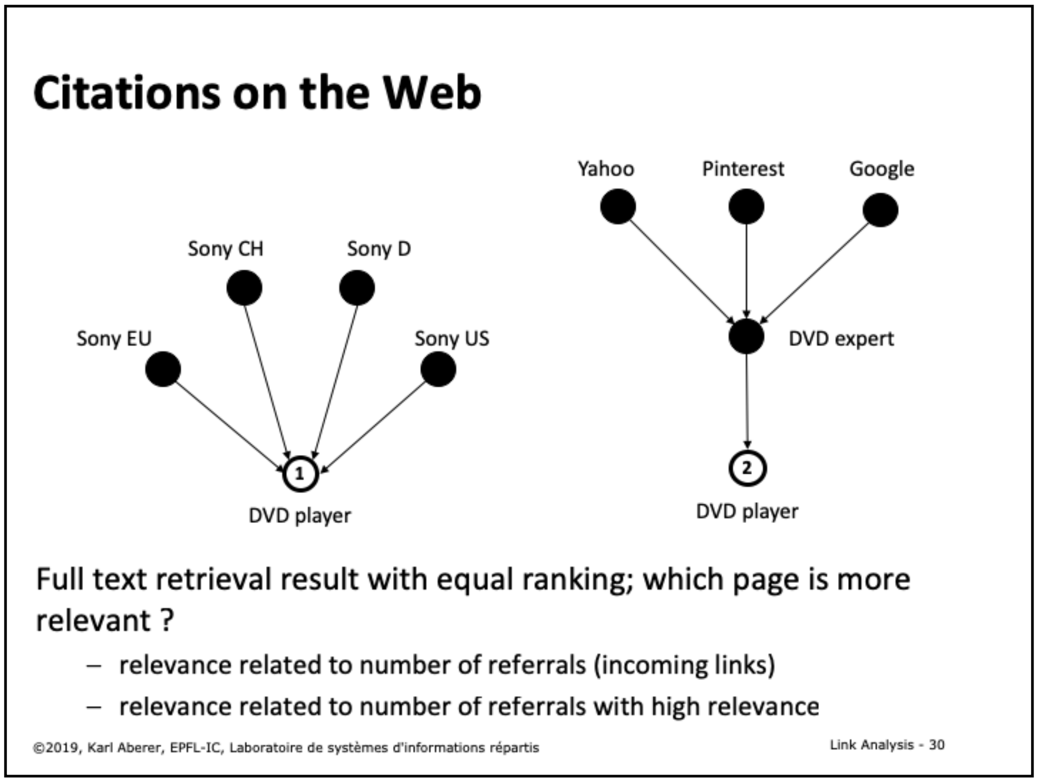
Bibliometry: analysis of citations in scientific publications

- Citation frequency: how important is a paper of author?
- Co-citation analysis: articles that co-cite the same articles are related
- Citation indexing: who is this author cited by?
- Impact factor: Authority of sources, such as journals

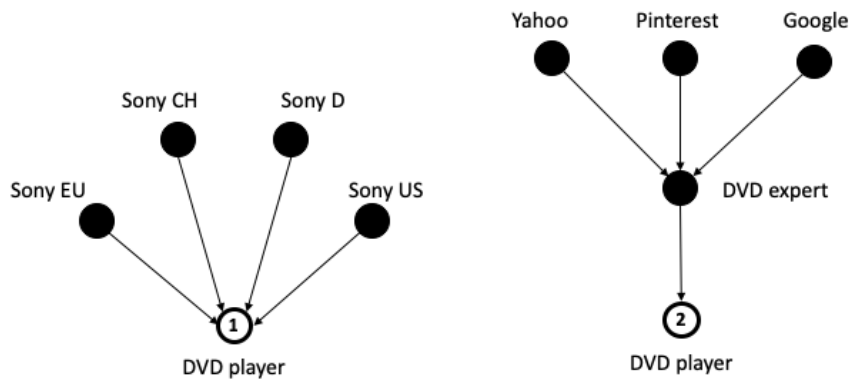
The use of links in order to evaluate the quality of information sources has a long tradition, in particular in science. The scientific discipline of bibliometry is fully devoted to the problem of evaluating the quality of research through citation analysis. Different ideas can be exploited to that end:

- The frequency of citations to a paper, indicating how popular / visible it is
- Co-citation analysis in order to identify research working in related disciplines
- Citation indexing in order to explore the profile of researchers referring to an author (as indicator of both the discipline and the quality)
- Analysis of the authority of sources of scientific publications, e.g. journals, publishers, conferences. This measure can then in turn be used to weight the relevance of publications.

All these ideas could also be exploited for any other document collections that have references, in particular for Web document collections with hyperlinks.



Citations on the Web

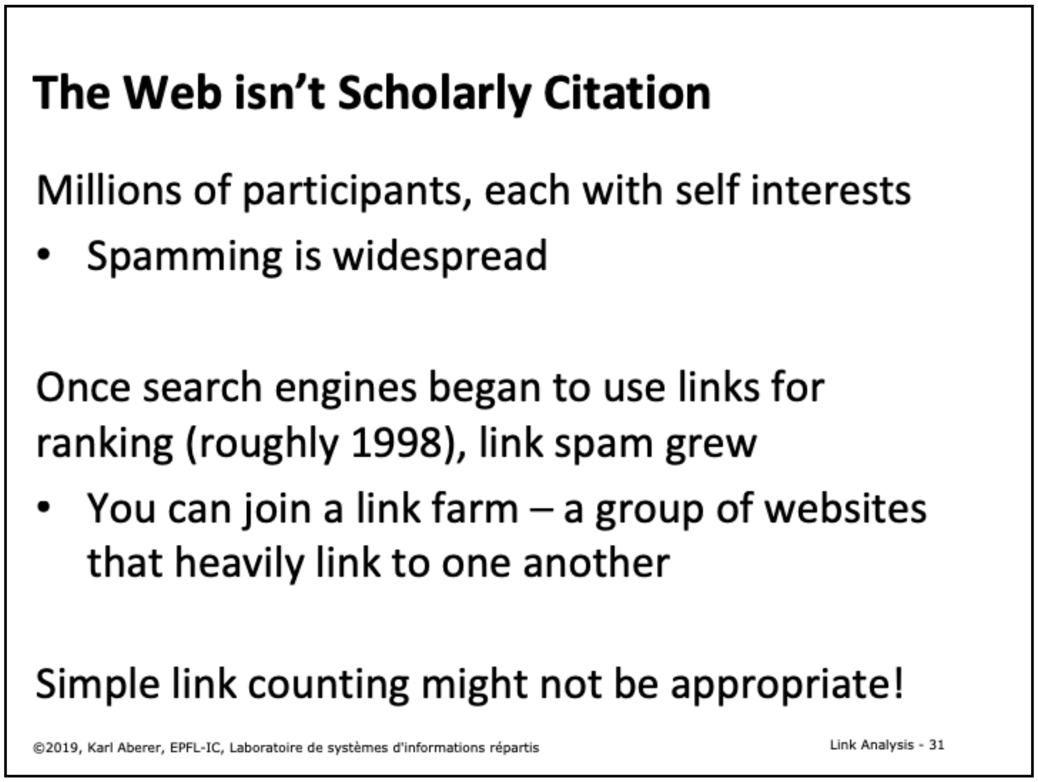


Full text retrieval result with equal ranking; which page is more relevant ?

- relevance related to number of referrals (incoming links)
- relevance related to number of referrals with high relevance

When retrieving documents from the Web, the link structure bears important information on the relevance of documents. Generally speaking, a document that is referred more often by other documents through Web links, is likely to be of higher interest and therefore relevance. So a possibility to rank documents is considering the number of incoming links. Doing this allows to distinguish documents that otherwise would be ranked equally or similarly when relying on text retrieval alone.

However, when doing this, also the importance of documents having a link to the document to be ranked may be different. Therefore not only counting then number of incoming links, but also weighing the links by the relevance of documents that contain these links appears to be appropriate. The same reasoning of course again applies then for evaluating the relevance of documents pointing to the document and so forth.



The Web isn't Scholarly Citation

Millions of participants, each with self interests

- Spamming is widespread

Once search engines began to use links for ranking (roughly 1998), link spam grew

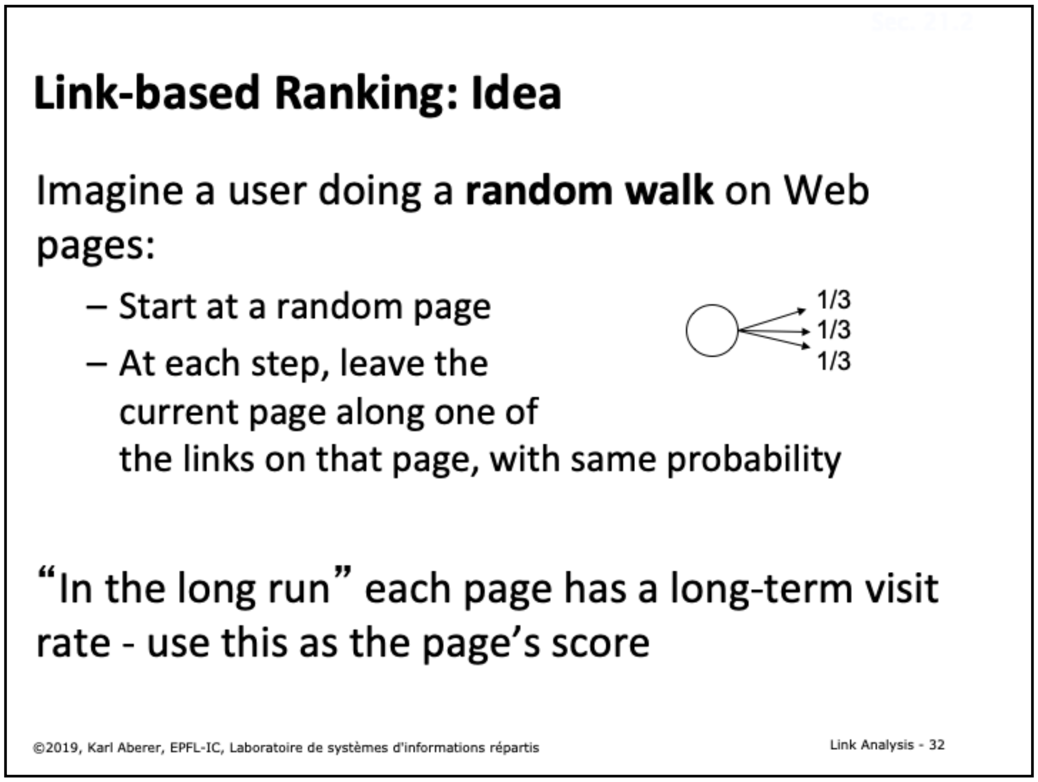
- You can join a link farm – a group of websites that heavily link to one another

Simple link counting might not be appropriate!

©2019, Karl Aberer, EPFL-IC, Laboratoire de systèmes d'informations répartis

Link Analysis - 31

In the web, analysis of links servers not only (or necessarily) the purpose of finding authorative sources, such as in science. It also and in particular can help to fight the aforementioned problem of link spamming. When using such methods, such as in any arms race, of course counter-measures have been devised by the adversaries. These resulted in the creation of link farms, that try to boost the relevance of Web pages by creating many (artificial) links to them. So just counting incoming links is probably not such a good idea in order to evaluate the quality of a Web page.

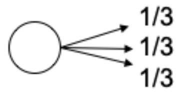


Sec. 21.2

Link-based Ranking: Idea

Imagine a user doing a **random walk** on Web pages:

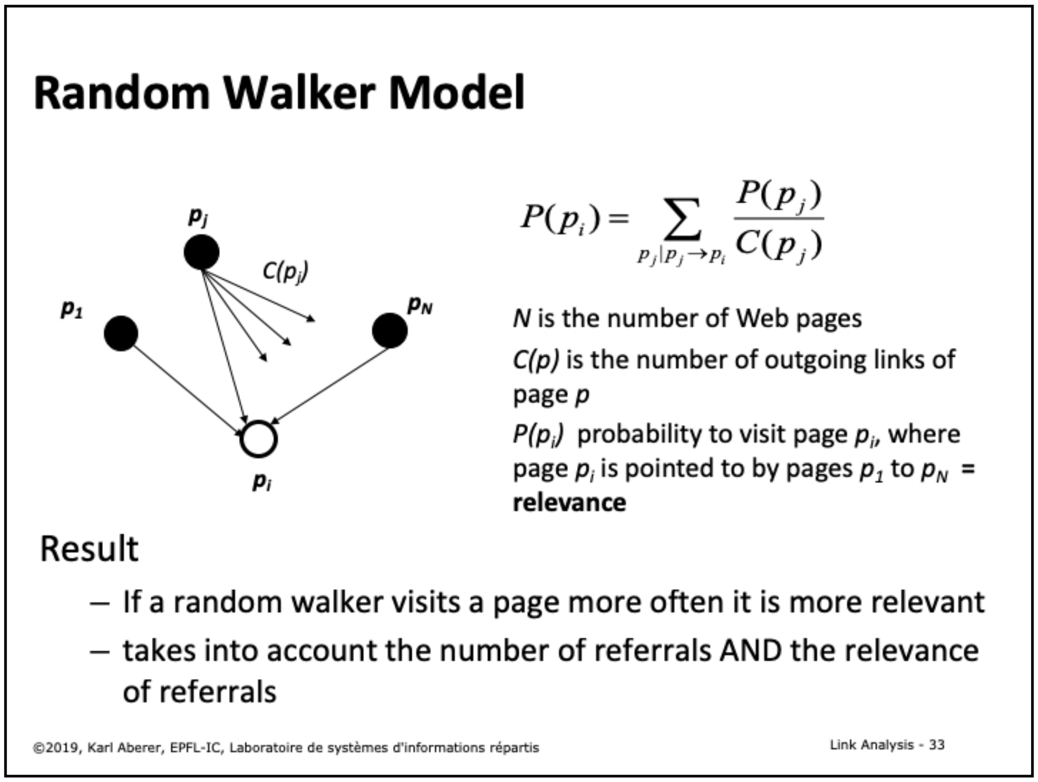
- Start at a random page
- At each step, leave the current page along one of the links on that page, with same probability



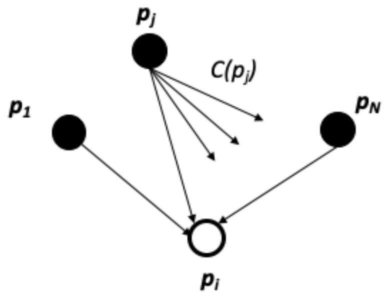
"In the long run" each page has a long-term visit rate - use this as the page's score

The fundamental idea of link-based scoring while attempting to fight spam is based on the concept of a random walker on the Web. Assume a random walker would visit a Web page. Then it would follow each f the outgoing links with equal probability. If walking for a very long time in the Web, it would have a certain fraction of visits it passes by every page.

One of the consequences of this model would be that pages that have few in-links, would be relatively infrequently visited. Since link farms usually have not many links pointing to them, in this way their influence in terms of link spamming would be moderated.



Random Walker Model



$$P(p_i) = \sum_{p_j | p_j \to p_i} \frac{P(p_j)}{C(p_j)}$$

N is the number of Web pages

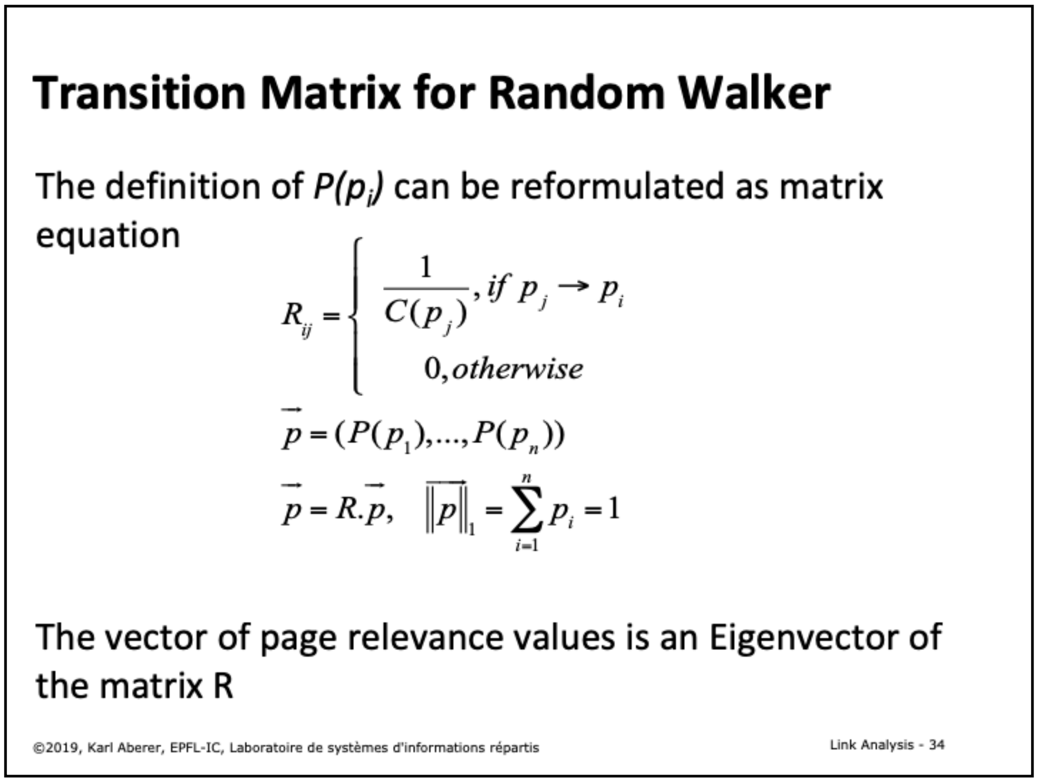
C(p) is the number of outgoing links of page *p*

P(p$_i$) probability to visit page *p$_i$*, where page *p$_i$* is pointed to by pages *p$_1$* to *p$_N$* = **relevance**

Result

- If a random walker visits a page more often it is more relevant
- takes into account the number of referrals AND the relevance of referrals

Here we describe the random walker model more formally. The long term probability of visiting a page, depends on the long-term probability of having visited a page that is referring to the page. Thus the formulation of the random walker model becomes recursive.



Transition Matrix for Random Walker

The definition of $P(p_i)$ can be reformulated as matrix equation

$$R_{ij} = \begin{cases} \frac{1}{C(p_j)}, if\ p_j \to p_i \\ 0, otherwise \end{cases}$$

$$\vec{p} = (P(p_1), \ldots, P(p_n))$$

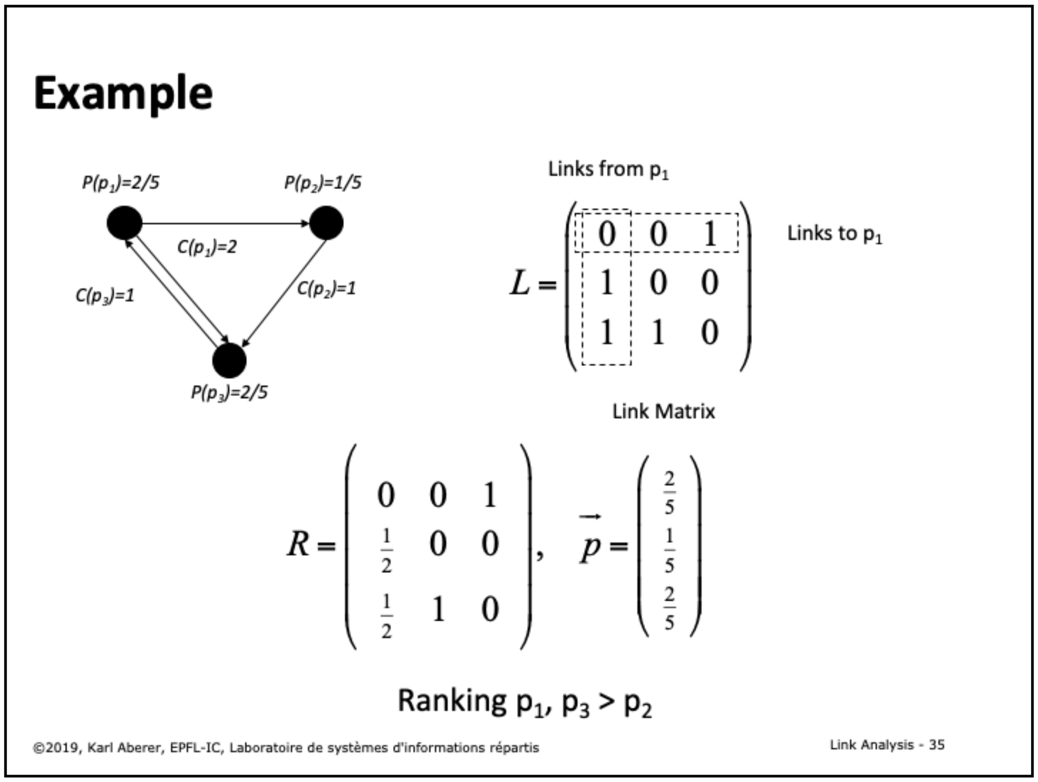
$$\vec{p} = R.\vec{p}, \quad \|\vec{p}\|_1 = \sum_{i=1}^{n} p_i = 1$$

The vector of page relevance values is an Eigenvector of the matrix R

©2019, Karl Aberer, EPFL-IC, Laboratoire de systèmes d'informations répartis

Link Analysis - 34

In order to determine the solution to the recursive formulation of the probabilities of a random walker to visit a page, we introduce a transition probability matrix R, which captures the probability of transitioning from one page to another. We also require that the long-term probabilities of visiting a page add up to 1. With this representation the long-term visiting probabilities become the Eigenvector of matrix R. More precisely, they are the Eigenvector with the largest Eigenvalue.



Example

P(p₁)=2/5
P(p₂)=1/5
C(p₁)=2
C(p₃)=1
C(p₂)=1
P(p₃)=2/5

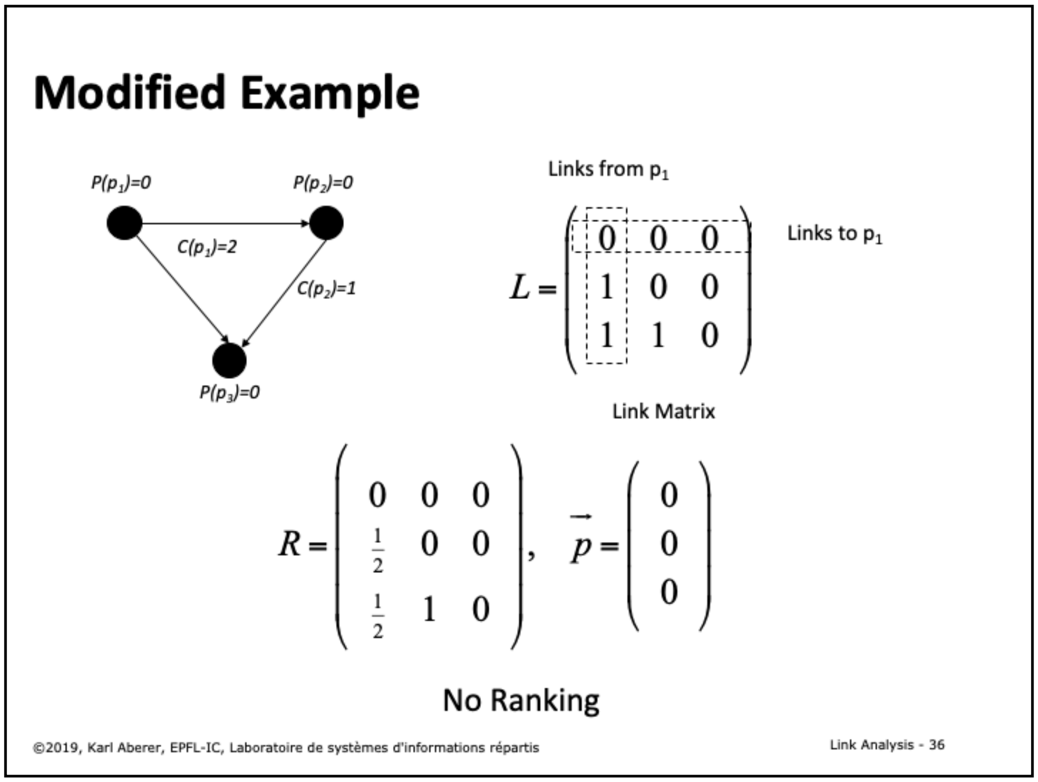
$$\begin{array}{cc} & \text{Links from } p_1 \\ L = & \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \quad \text{Links to } p_1 \end{array}$$

Link Matrix

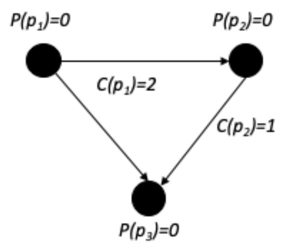
$$R = \begin{pmatrix} 0 & 0 & 1 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \end{pmatrix}, \quad \vec{p} = \begin{pmatrix} \frac{2}{5} \\ \frac{1}{5} \\ \frac{2}{5} \end{pmatrix}$$

Ranking p_1, $p_3 > p_2$

This example illustrates the computation of the probabilities for visiting a specific Web page. The values C(p_i) correspond to the transition probabilities. They can be derived from the link matrix. The link matrix is defined as L_{ij}=1 if there is a link from p_j to p_i. The link matrix is normalized by the outdegree, by dividing the values in the columns by the sum of the values found in the column, resulting in Matrix R. The probability of a random walker being in a node is then obtained from the Eigenvector of this matrix.



Modified Example



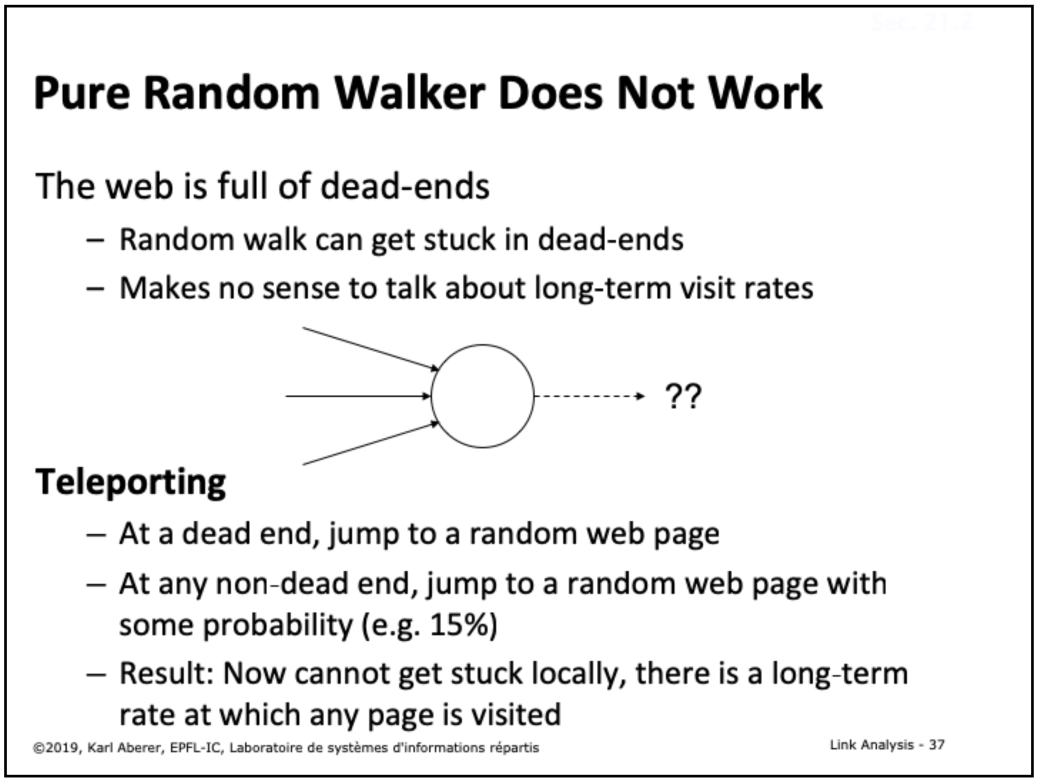
$$
L = \begin{array}{cc} \text{Links from } p_1 & \\ \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} & \text{Links to } p_1 \end{array}
$$

Link Matrix

$$
R = \begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \end{pmatrix}, \quad \vec{p} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}
$$

No Ranking

This example illustrates the problem of dead ends. We see that there exists a node p_3 that is a "sink of rank". Any random walk ends up in this sink, and therefore the other nodes do not receive any ranking weight. Consequently also the rank of sink does not. Therefore the only solution to the equation p=Rp is the zero vector.

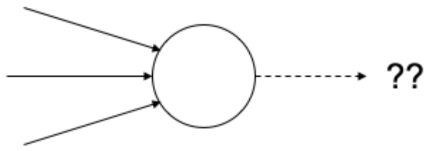


Sec. 21.2

Pure Random Walker Does Not Work

The web is full of dead-ends

- Random walk can get stuck in dead-ends
- Makes no sense to talk about long-term visit rates

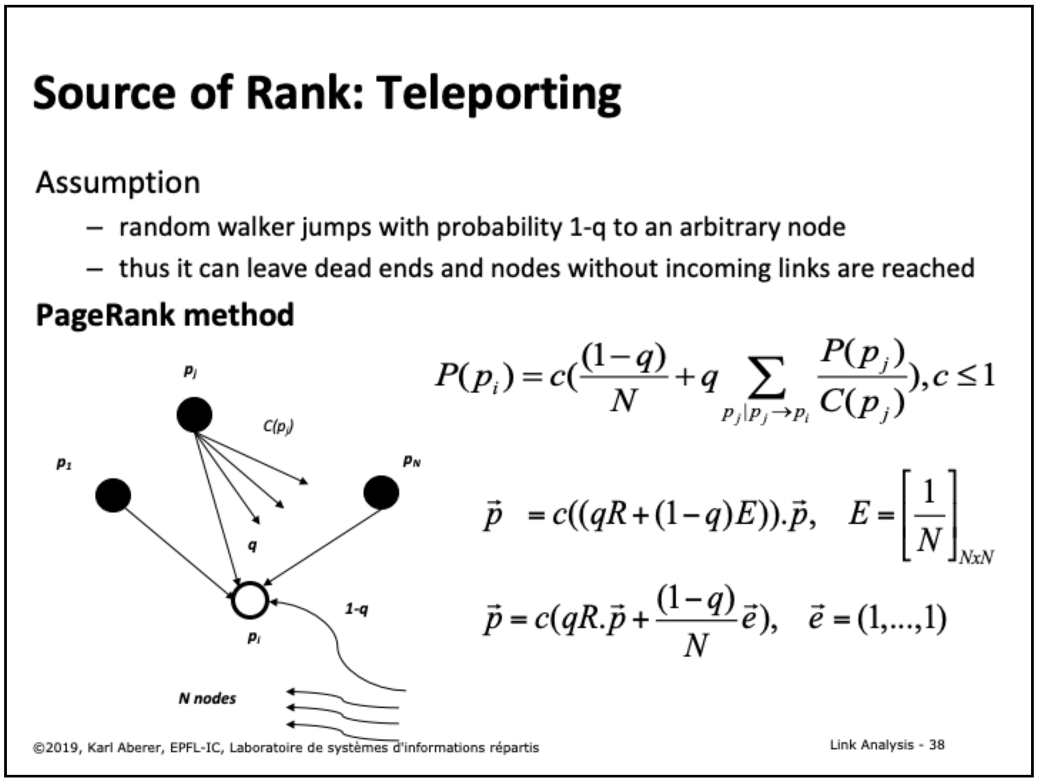


Teleporting

- At a dead end, jump to a random web page
- At any non-dead end, jump to a random web page with some probability (e.g. 15%)
- Result: Now cannot get stuck locally, there is a long-term rate at which any page is visited

A practical problem with the random walker is the fact that there exist Web pages that have no outgoing links. Thus the random walker would get stuck. To fix this, the concept of teleporting is introduced, where the random walker jumps to any randomly selected Web page. In case of arriving at a dead end, it jumps in any case, otherwise with a given probability instead of following an outgoing link.

Another problem are pages that have no incoming links: they would never be reached by the random walker, and the weight that they could provide to other pages would not be considered. Also this problem is solved by teleporting.

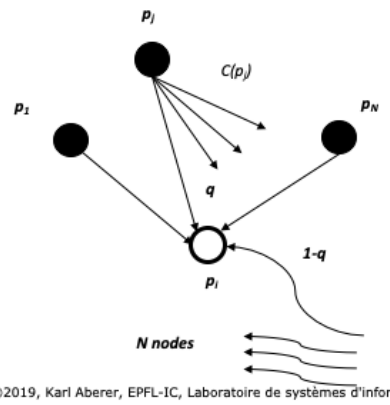


Source of Rank: Teleporting

Assumption

- random walker jumps with probability 1-q to an arbitrary node
- thus it can leave dead ends and nodes without incoming links are reached

PageRank method



$$P(p_i) = c\left(\frac{(1-q)}{N} + q \sum_{p_j \mid p_j \to p_i} \frac{P(p_j)}{C(p_j)}\right), c \leq 1$$

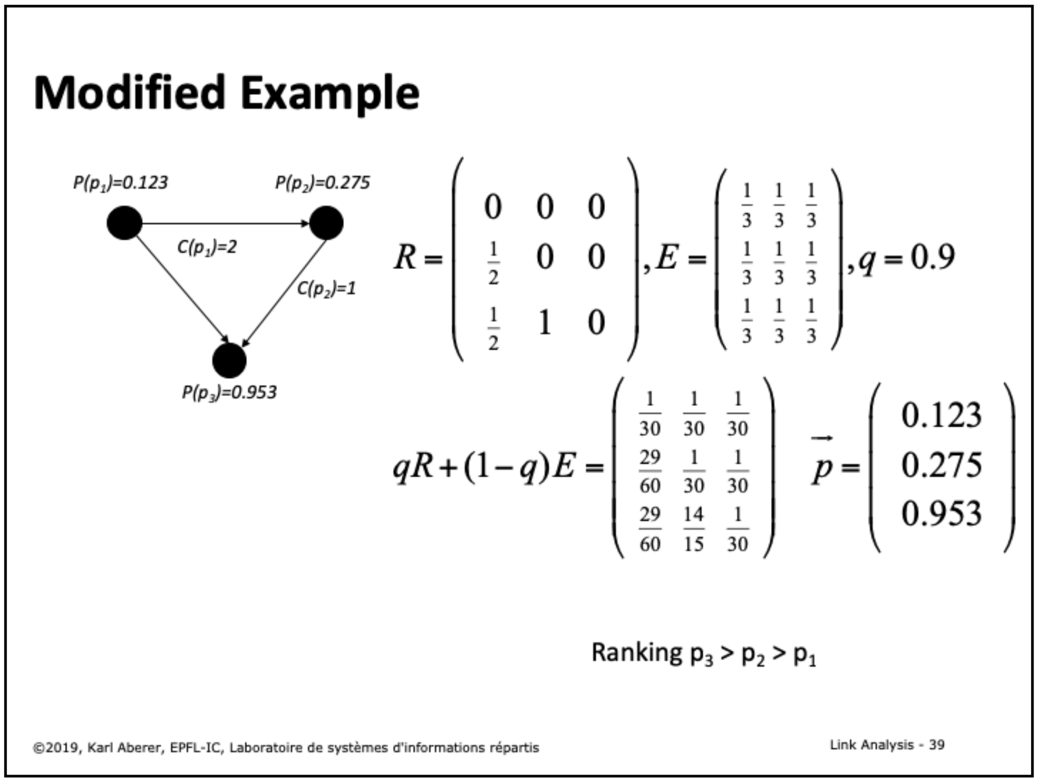
$$\vec{p} = c((qR + (1-q)E)).\vec{p}, \quad E = \left[\frac{1}{N}\right]_{NxN}$$

$$\vec{p} = c(qR.\vec{p} + \frac{(1-q)}{N}\vec{e}), \quad \vec{e} = (1,...,1)$$

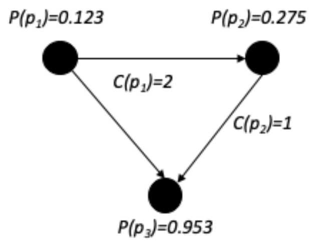
©2019, Karl Aberer, EPFL-IC, Laboratoire de systèmes d'informations répartis

Link Analysis - 38

To avoid the previously described problem, we add a "source of rank". The idea is that a random walker in each step can, rather than following a link, jump to any page with probability 1-q. Therefore the random walker can leave pages that have no outgoing links and also pages without incoming links can be reached by the random walk and give weight to other pages. In the mathematical formulation of the random walk this is resulting in an additional a term for the source of rank. Since at each step the random walker makes a jump with probability 1-q and any of the N pages is reached with the same probability the additional term is (1-q)/N. Reformulating this again as matrix equation requires adding a NxN Matrix E with all entries being 1/N. This is equivalent to saying that with probability 1/N transitions among any pairs of nodes (including transition from a node to itself) are performed. Since the vector p has norm 1, i.e., the sum of the components is exactly 1, E.p=e, where e is the unit vector, the matrix equation can be reformulated in the second form shown below. The method described is called PageRank and is used by Google. By modifying the values of the matrix E also a priori knowledge about the relative importance of pages can be added to the algorithm.



Modified Example

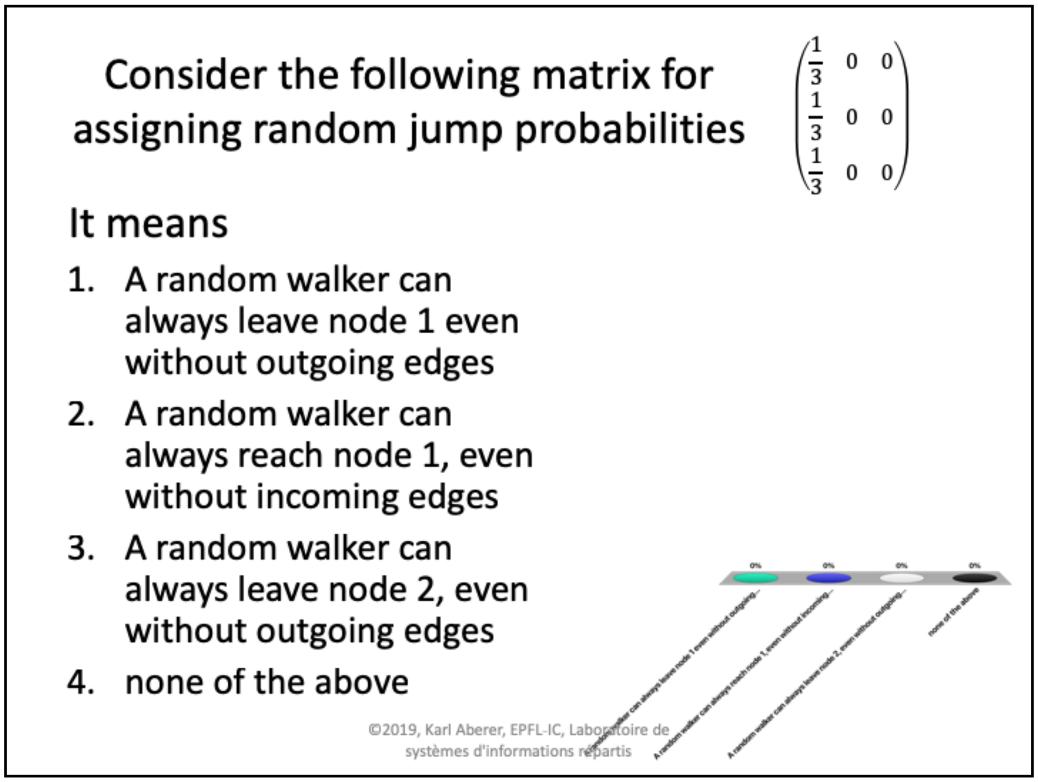


$$R=\begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \end{pmatrix}, E=\begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}, q=0.9$$

$$qR+(1-q)E=\begin{pmatrix} \frac{1}{30} & \frac{1}{30} & \frac{1}{30} \\ \frac{29}{60} & \frac{1}{30} & \frac{1}{30} \\ \frac{29}{60} & \frac{14}{15} & \frac{1}{30} \end{pmatrix} \quad \vec{p}=\begin{pmatrix} 0.123 \\ 0.275 \\ 0.953 \end{pmatrix}$$

Ranking $p_3 > p_2 > p_1$

With the modification of rank computation using a source of rank, we obtain for our example again a non-trivial ranking which appears to be appropriate.

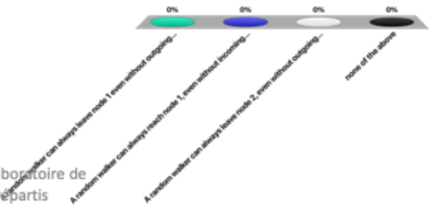


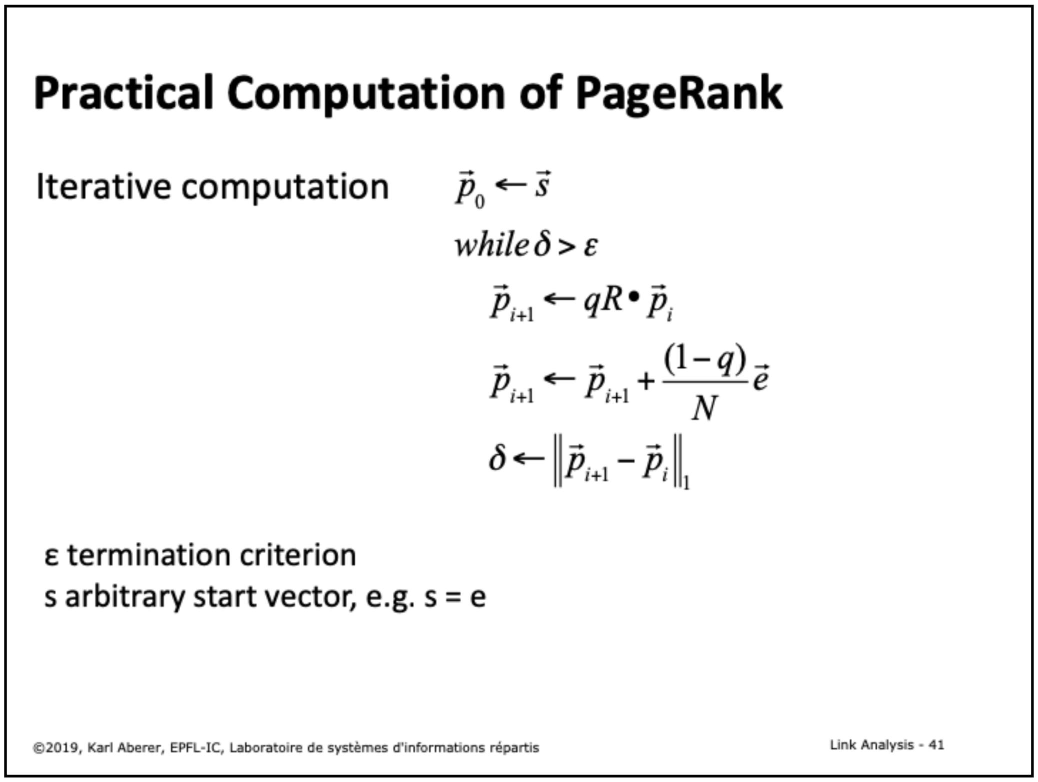
Consider the following matrix for assigning random jump probabilities

$$\begin{pmatrix} \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & 0 & 0 \end{pmatrix}$$

It means

1. A random walker can always leave node 1 even without outgoing edges
2. A random walker can always reach node 1, even without incoming edges
3. A random walker can always leave node 2, even without outgoing edges
4. none of the above





Practical Computation of PageRank

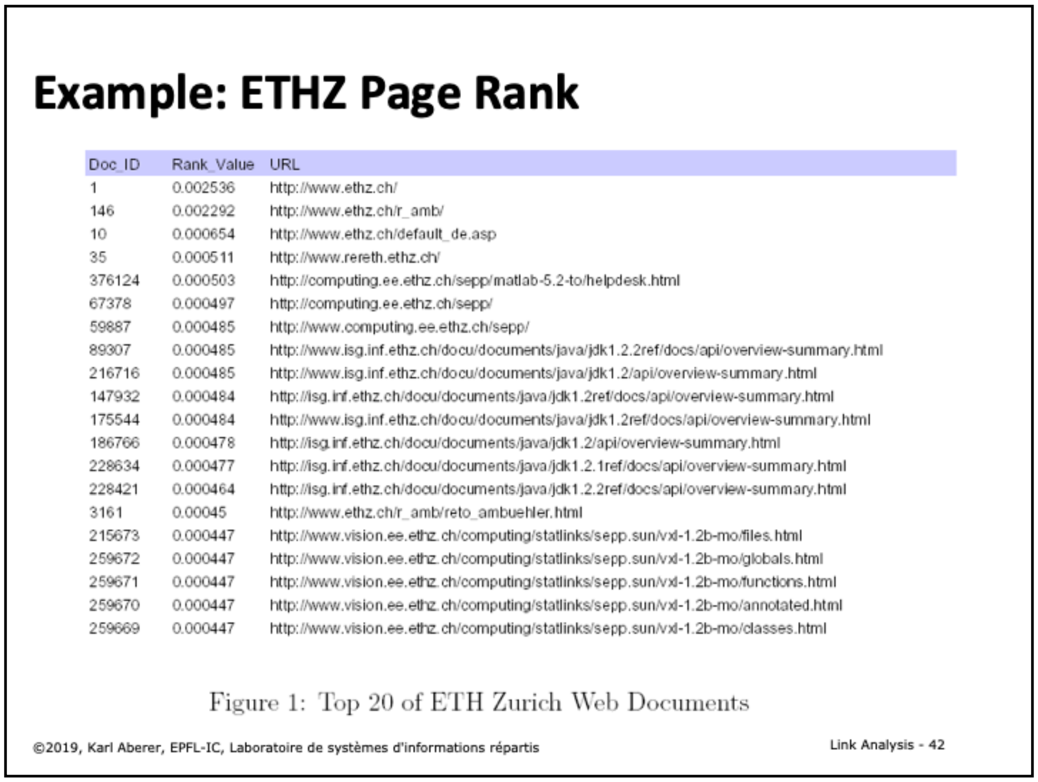
Iterative computation

```
$\vec{p}_0 \leftarrow \vec{s}$
$while\, \delta > \varepsilon$
    $\vec{p}_{i+1} \leftarrow qR \bullet \vec{p}_i$
    $\vec{p}_{i+1} \leftarrow \vec{p}_{i+1} + \frac{(1-q)}{N}\vec{e}$
    $\delta \leftarrow \left\| \vec{p}_{i+1} - \vec{p}_i \right\|_1$
```

ε termination criterion

s arbitrary start vector, e.g. s = e

For the practical computation of the PageRank ranking an iterative approach can be used. It is derived from the second form of the formulation of the visiting probabilities of the random walker that we have given. The vector e used to add a source of rank has not necessarily to assign uniform weights to all pages, but might reflect itself a ranking of Web pages.

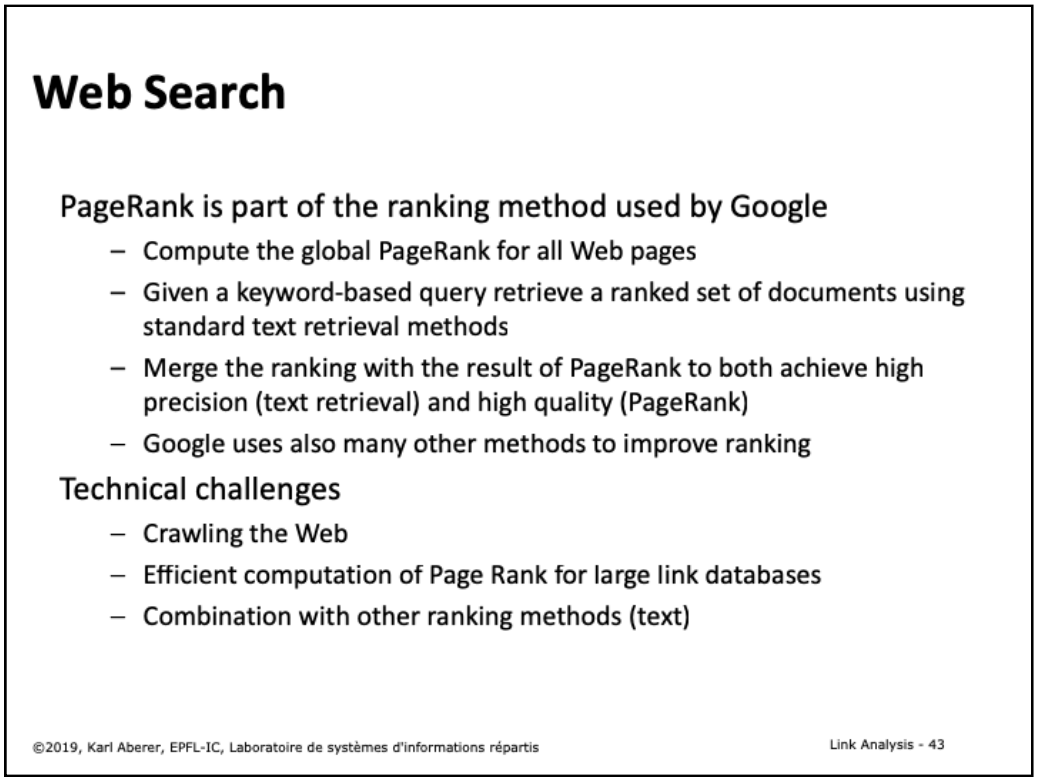


Example: ETHZ Page Rank

Doc_ID	Rank_Value	URL
1	0.002536	http://www.ethz.ch/
146	0.002292	http://www.ethz.ch/r_amb/
10	0.000654	http://www.ethz.ch/default_de.asp
35	0.000511	http://www.rereth.ethz.ch/
376124	0.000503	http://computing.ee.ethz.ch/sepp/matlab-5.2-to/helpdesk.html
67378	0.000497	http://computing.ee.ethz.ch/sepp/
59887	0.000485	http://www.computing.ee.ethz.ch/sepp/
89307	0.000485	http://www.isg.inf.ethz.ch/docu/documents/java/jdk1.2.2ref/docs/api/overview-summary.html
216716	0.000485	http://www.isg.inf.ethz.ch/docu/documents/java/jdk1.2/api/overview-summary.html
147932	0.000484	http://isg.inf.ethz.ch/docu/documents/java/jdk1.2ref/docs/api/overview-summary.html
175544	0.000484	http://www.isg.inf.ethz.ch/docu/documents/java/jdk1.2ref/docs/api/overview-summary.html
186766	0.000478	http://isg.inf.ethz.ch/docu/documents/java/jdk1.2/api/overview-summary.html
228634	0.000477	http://isg.inf.ethz.ch/docu/documents/java/jdk1.2.1ref/docs/api/overview-summary.html
228421	0.000464	http://isg.inf.ethz.ch/docu/documents/java/jdk1.2.2ref/docs/api/overview-summary.html
3161	0.00045	http://www.ethz.ch/r_amb/reto_ambuehler.html
215673	0.000447	http://www.vision.ee.ethz.ch/computing/statlinks/sepp.sun/vxl-1.2b-mo/files.html
259672	0.000447	http://www.vision.ee.ethz.ch/computing/statlinks/sepp.sun/vxl-1.2b-mo/globals.html
259671	0.000447	http://www.vision.ee.ethz.ch/computing/statlinks/sepp.sun/vxl-1.2b-mo/functions.html
259670	0.000447	http://www.vision.ee.ethz.ch/computing/statlinks/sepp.sun/vxl-1.2b-mo/annotated.html
259669	0.000447	http://www.vision.ee.ethz.ch/computing/statlinks/sepp.sun/vxl-1.2b-mo/classes.html

Figure 1: Top 20 of ETH Zurich Web Documents

These are the top documents from the PageRank ranking of all Web pages at ETHZ (Data from 2001). It is interesting to see that documents related to Java documentation receive high ranking values. This is related to the fact that these documents have many internal cross-references.



Web Search

PageRank is part of the ranking method used by Google

- Compute the global PageRank for all Web pages
- Given a keyword-based query retrieve a ranked set of documents using standard text retrieval methods
- Merge the ranking with the result of PageRank to both achieve high precision (text retrieval) and high quality (PageRank)
- Google uses also many other methods to improve ranking

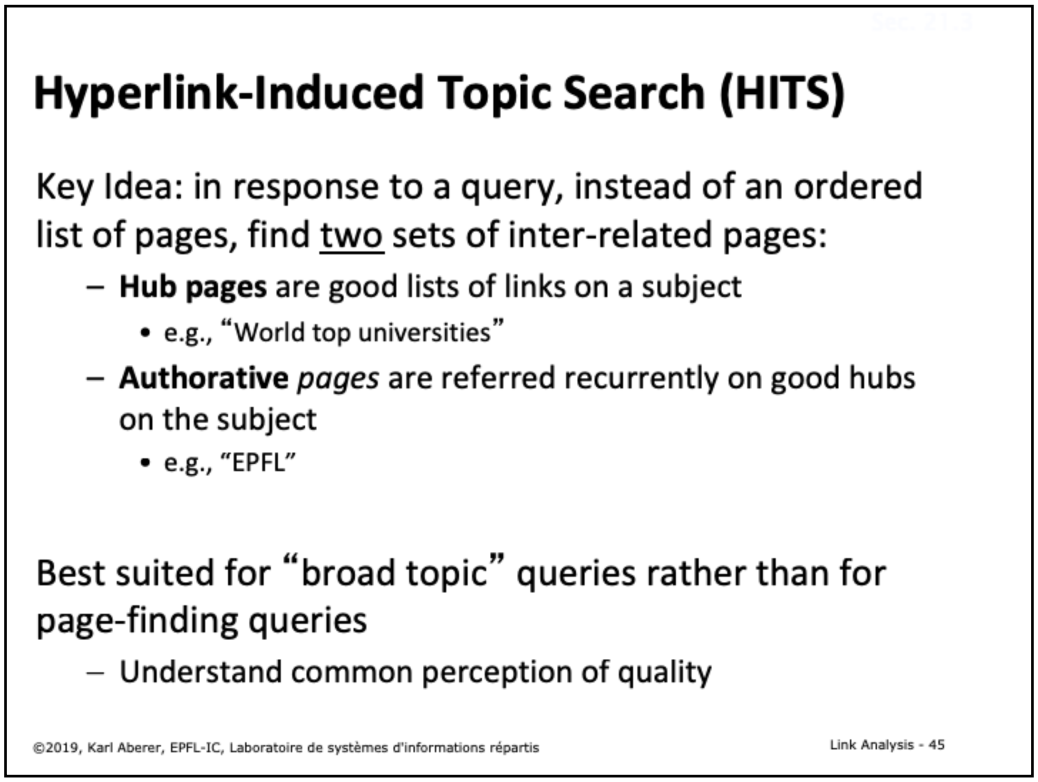
Technical challenges

- Crawling the Web
- Efficient computation of Page Rank for large link databases
- Combination with other ranking methods (text)

PageRank is used as one metrics to rank result documents in Google. Essentially Google uses text retrieval methods to retrieve relevant documents and then applies PageRank to create a more appropriate ranking. Google uses also many other methods to improve ranking, e.g., by giving different weights to different parts of Web documents and user relevance feedbacks. For example, title elements are given higher weight. The details of the ranking methods are trade secrets of the Web search engine providers.

Other issues that Web search engines have to deal with, are crawling the Web, which requires techniques that can explore the Web without revisiting pages too frequently. Also the enormous size of the document and link database poses implementation challenges in order to keep the ranking computations scalable. One of the outcomes of solving these challenges are the recent "cloud computing" infrastructures, which are large-scale computing clusters constructed from commodity hardware.

LINK-BASED RANKING: HITS



Sec. 21.3

Hyperlink-Induced Topic Search (HITS)

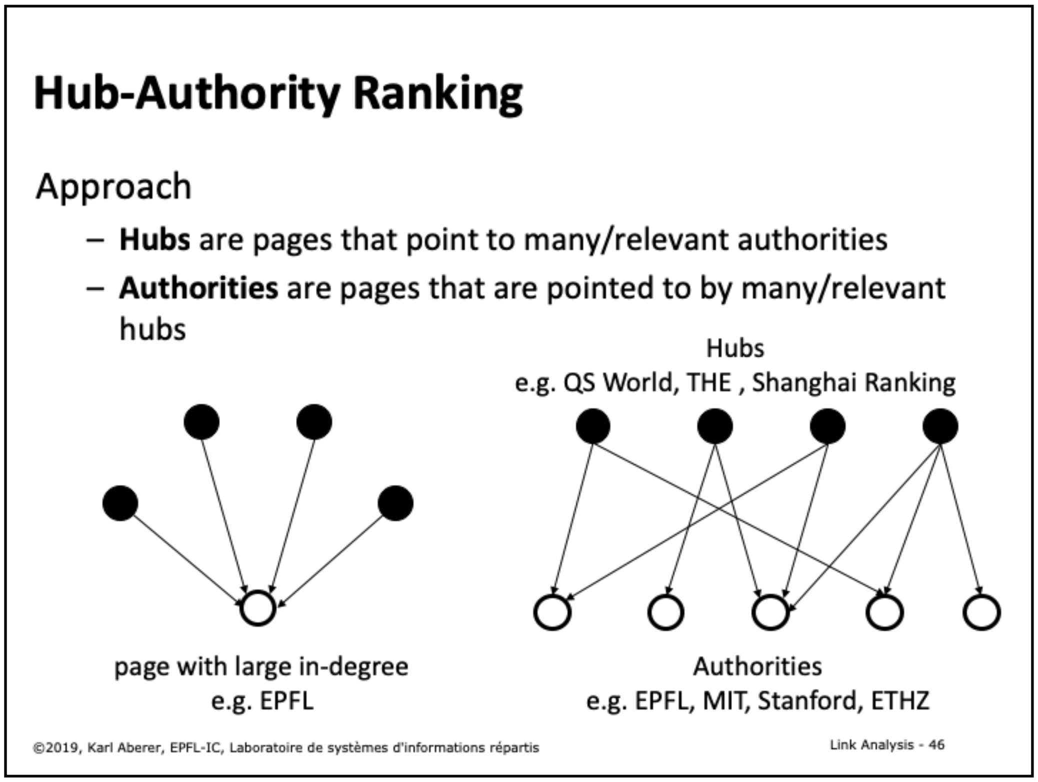
Key Idea: in response to a query, instead of an ordered list of pages, find two sets of inter-related pages:

- **Hub pages** are good lists of links on a subject
  - e.g., "World top universities"
- **Authorative** *pages* are referred recurrently on good hubs on the subject
  - e.g., "EPFL"

Best suited for "broad topic" queries rather than for page-finding queries

- Understand common perception of quality

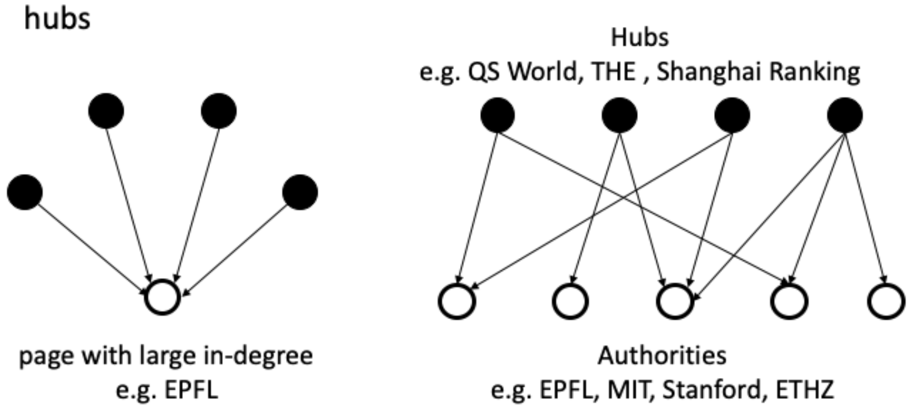
Hub-Authority ranking identifies not only pages that have a high authority, as measured by the number of incoming links, but also pages that have a substantial "referential" value, having many outgoing links (to pages of high importance). Different to the PageRank algorithm this technique has been originally proposed to post-process query results (rather than to rank pages from the complete Web graph). It can be used as a add-on to existing search engines, but as well as an alternative to the PageRank method.



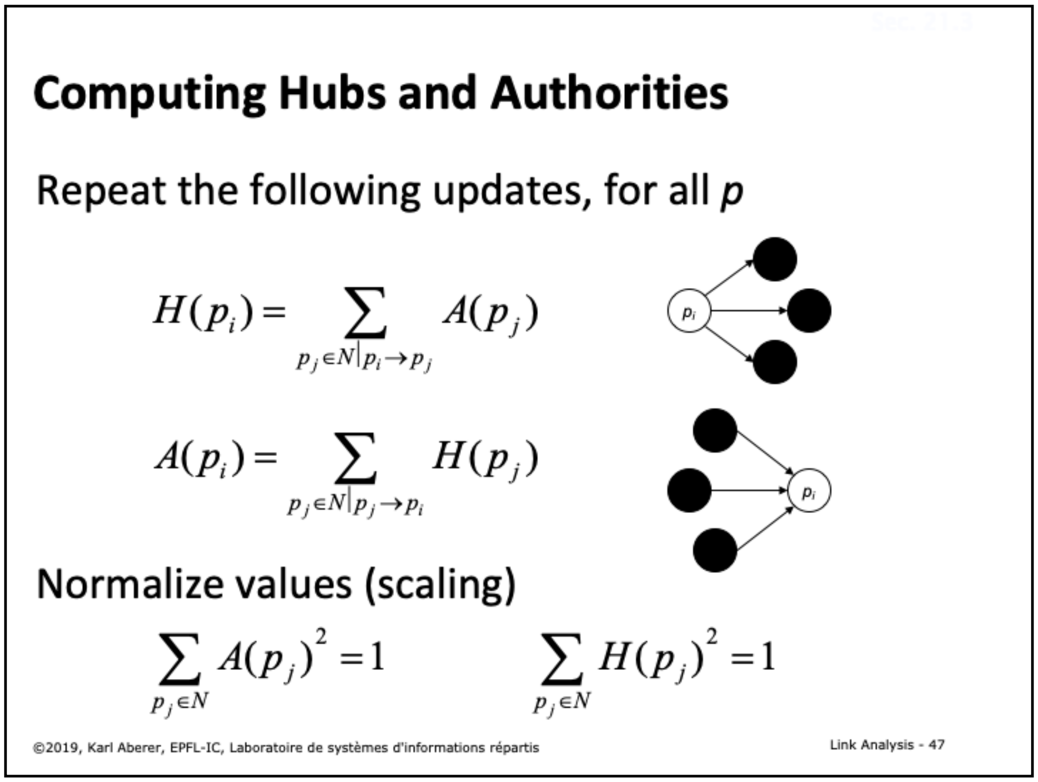
Hub-Authority Ranking

Approach

- **Hubs** are pages that point to many/relevant authorities
- **Authorities** are pages that are pointed to by many/relevant hubs


Hubs
e.g. QS World, THE , Shanghai Ranking
page with large in-degree
e.g. EPFL
Authorities
e.g. EPFL, MIT, Stanford, ETHZ

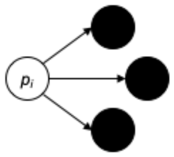
In order to realize the idea of distinguish authorities from hubs a simple approach is taken. Hub pages are consider as those that are referred a lot by authority pages and vice versa. The example illustrates of how this would ideally separate authority pages, in the case of universities well known universities, from hub pages, such as university ranking and portal sites.



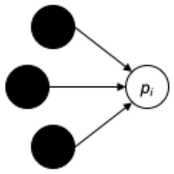
Computing Hubs and Authorities

Repeat the following updates, for all *p*

$$H(p_i) = \sum_{p_j \in N | p_i \to p_j} A(p_j)$$



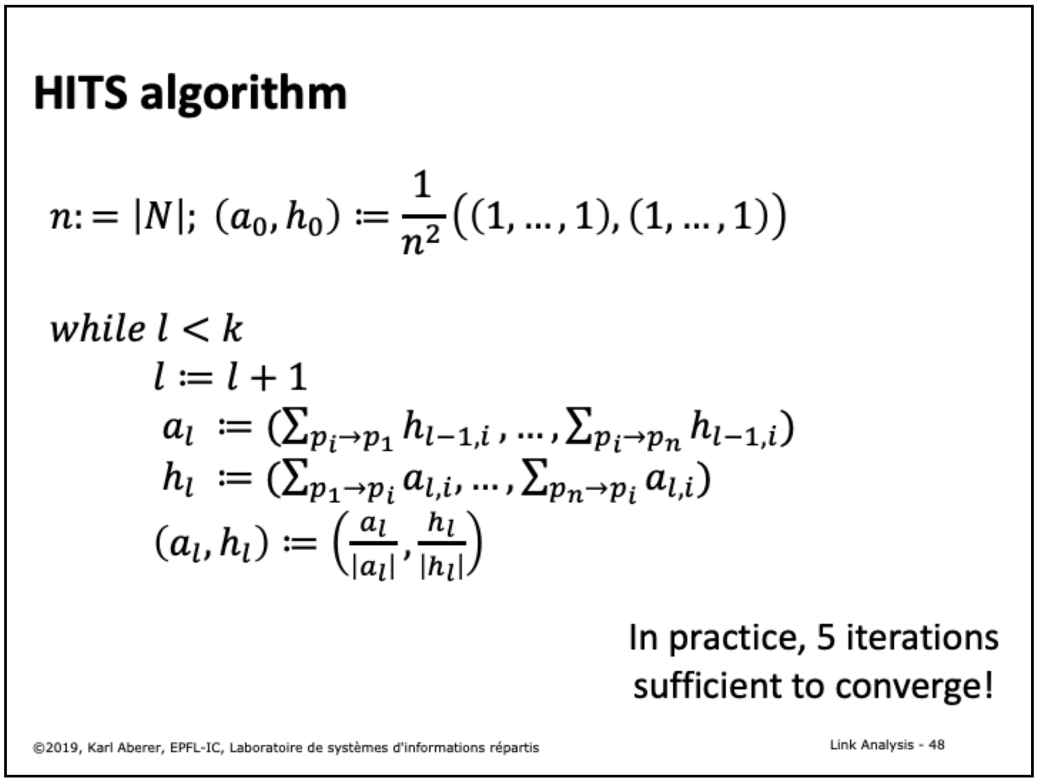
$$A(p_i) = \sum_{p_j \in N | p_j \to p_i} H(p_j)$$



Normalize values (scaling)

$$\sum_{p_j \in N} A(p_j)^2 = 1 \qquad \sum_{p_j \in N} H(p_j)^2 = 1$$

More precisely, the scores are recomputed by simply adding the scores of all incoming edges. For computing authority scores this is the hub scores and for hub scores the authority scores. In order to avoid that the scores grow continuously, they are rescaled to in each step, by normalizing the score vectors to one.



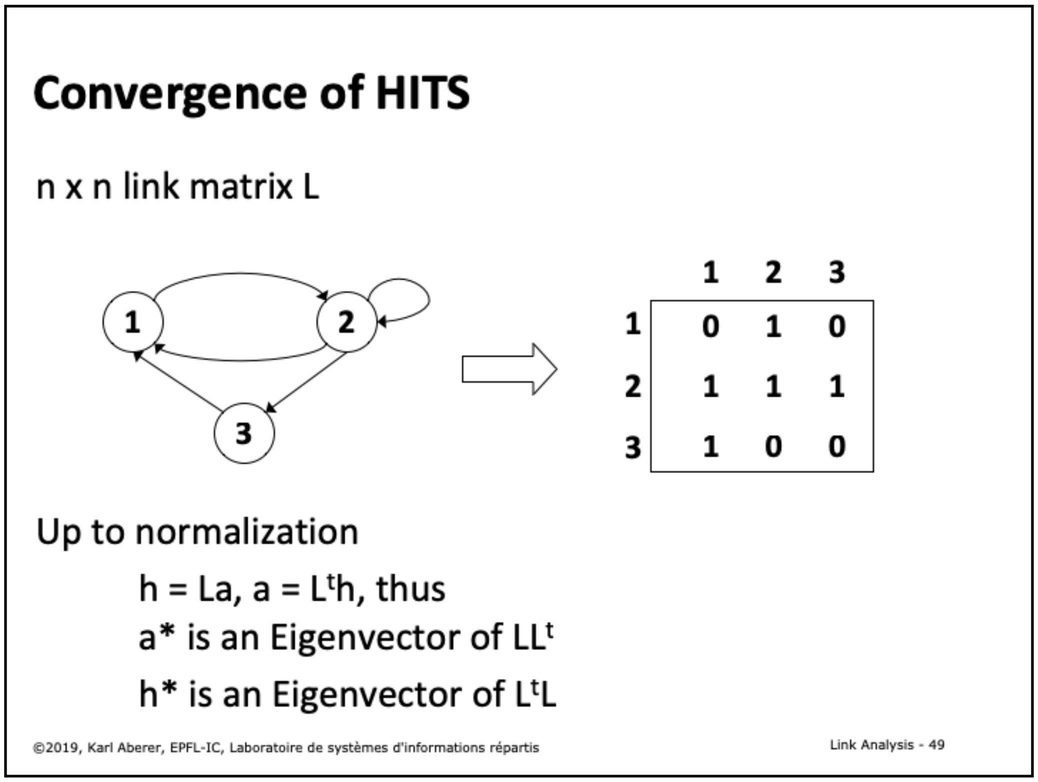
HITS algorithm

```
$n := |N|;\ (a_0, h_0) := \frac{1}{n^2}\big((1, \dots, 1), (1, \dots, 1)\big)$

$while\ l < k$
    $l := l + 1$
    $a_l := (\sum_{p_i \to p_1} h_{l-1,i}, \dots, \sum_{p_i \to p_n} h_{l-1,i})$
    $h_l := (\sum_{p_1 \to p_i} a_{l,i}, \dots, \sum_{p_n \to p_i} a_{l,i})$
    $(a_l, h_l) := \left(\frac{a_l}{|a_l|}, \frac{h_l}{|h_l|}\right)$
```

In practice, 5 iterations sufficient to converge!

As for PageRank the Hub/Authority values can be iteratively computed. x corresponds to the authority weight and y to the hub weight. At each iteration the values are renormalized to length 1.



Convergence of HITS

n x n link matrix L

$$\begin{tikzcd} 1 \arrow[rr, bend left] & & 2 \arrow[ll, bend left] \arrow[loop right] \arrow[dl] \\ & 3 \arrow[ul] & \end{tikzcd} \Rightarrow \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 1 & 0 \\ 2 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \end{array}$$

Up to normalization

$h = La$, $a = L^t h$, thus

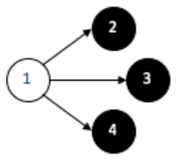
a^* is an Eigenvector of LL^t

h^* is an Eigenvector of L^tL

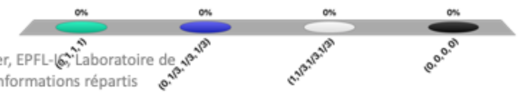
We can interpret the iterative algorithm for computing HITS in terms of computing Eigenvectors for the link matrix. If L is the link matrix then the computation of a from h is defined as a=Lh and the computation of h from a is defined as $h=L^ta$. Therefore a*, the authority vector obtained after convergence, is the principal Eigenvector of Matrix LL^t and h*, the hub vector obtained after convergence, is the principal Eigenvector of matrix L^tL.

Remains the question whether this algorithm always converges. The algorithm is a particular algorithm for computing eigenvectors: the *power iteration* method. It is proven to converge against the principal Eigenvalue. Important is the normalization of the vectors obtained in each step.

The authority values of this graph are



1. (0, 1, 1, 1)
2. (0, 1/3, 1/3, 1/3)
3. (1,1/3,1/3,1/3)
4. (0, 0, 0, 0)



Obtaining the Base Set

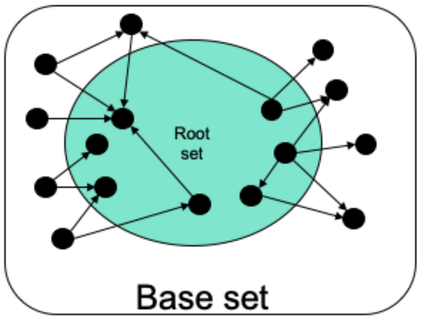
Given a query (e.g. “EPFL”), obtain all pages mentioning the query

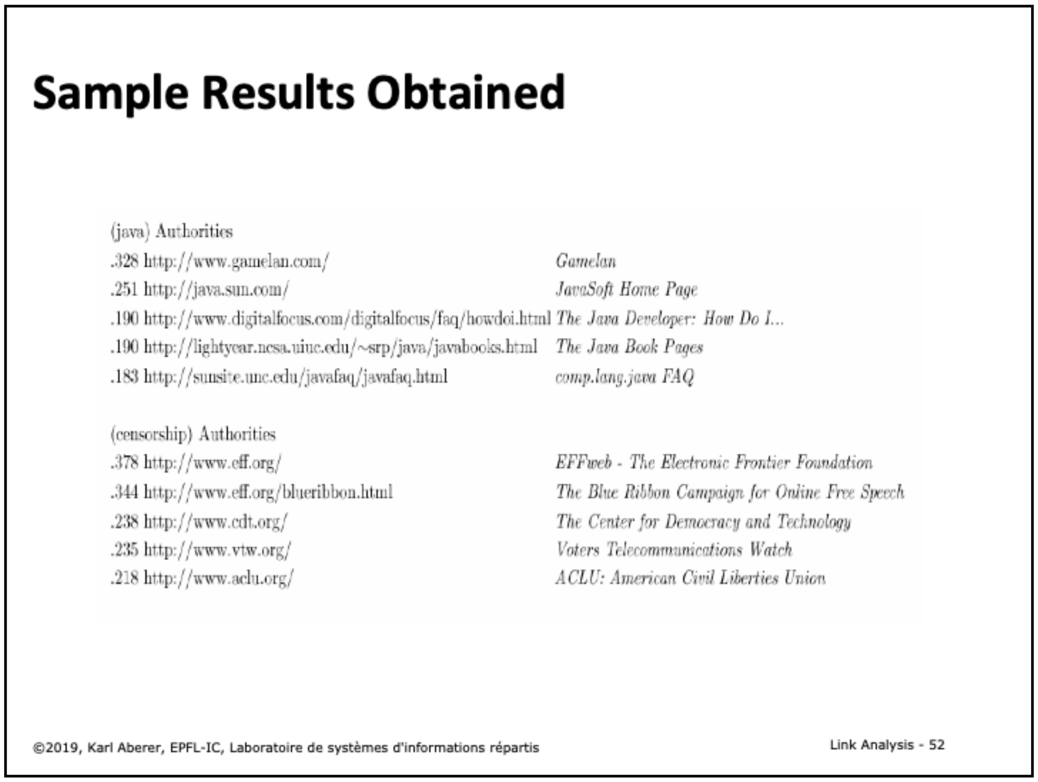
- call this the **root set** of pages.

Add page that either

- points to a page in the root set, or
- is pointed to by a page in the root set.

Use this set as **base set**

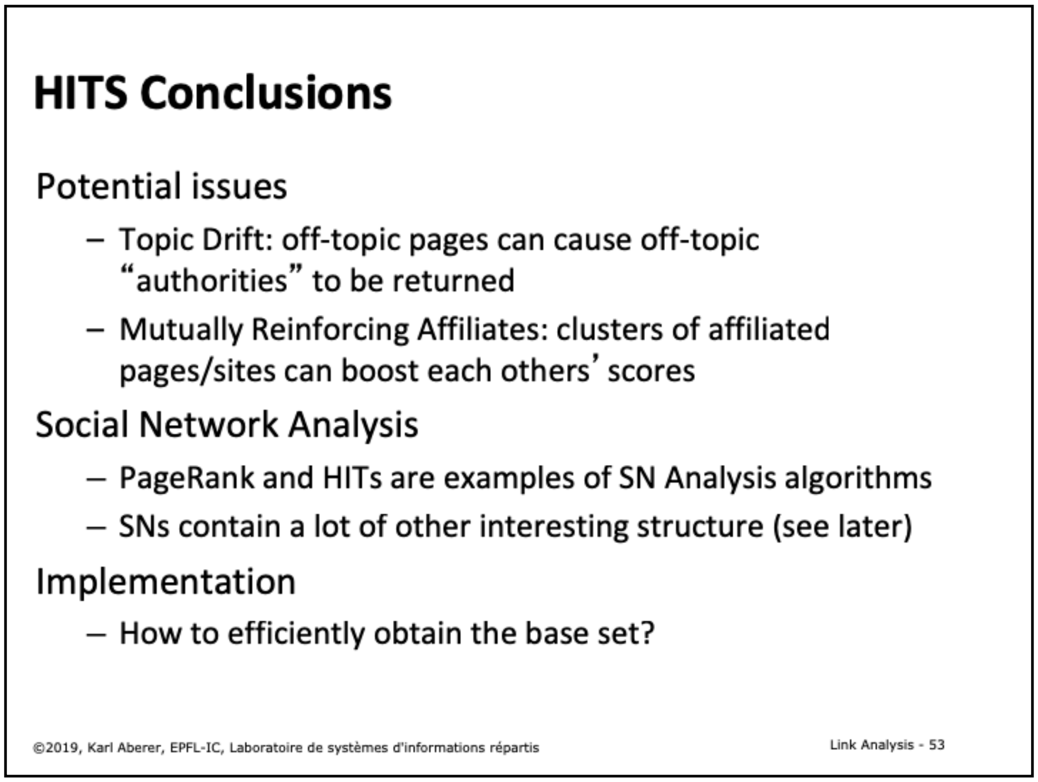




Sample Results Obtained

(java) Authorities	
.328 http://www.gamelan.com/	*Gamelan*
.251 http://java.sun.com/	*JavaSoft Home Page*
.190 http://www.digitalfocus.com/digitalfocus/faq/howdoi.html	*The Java Developer: How Do I...*
.190 http://lightyear.ncsa.uiuc.edu/~srp/java/javabooks.html	*The Java Book Pages*
.183 http://sunsite.unc.edu/javafaq/javafaq.html	*comp.lang.java FAQ*
(censorship) Authorities	
.378 http://www.eff.org/	*EFFweb - The Electronic Frontier Foundation*
.344 http://www.eff.org/blueribbon.html	*The Blue Ribbon Campaign for Online Free Speech*
.238 http://www.cdt.org/	*The Center for Democracy and Technology*
.235 http://www.vtw.org/	*Voters Telecommunications Watch*
.218 http://www.aclu.org/	*ACLU: American Civil Liberties Union*

These are two example results that have been obtained by applying this method. (It is interesting to compare those to the ones you would obtain by using a search engine alone). In particular Gamelan will not show up in the result, whereas the Java Sun page is usually among the top ranked.



HITS Conclusions

Potential issues

- Topic Drift: off-topic pages can cause off-topic "authorities" to be returned
- Mutually Reinforcing Affiliates: clusters of affiliated pages/sites can boost each others' scores

Social Network Analysis

- PageRank and HITs are examples of SN Analysis algorithms
- SNs contain a lot of other interesting structure (see later)

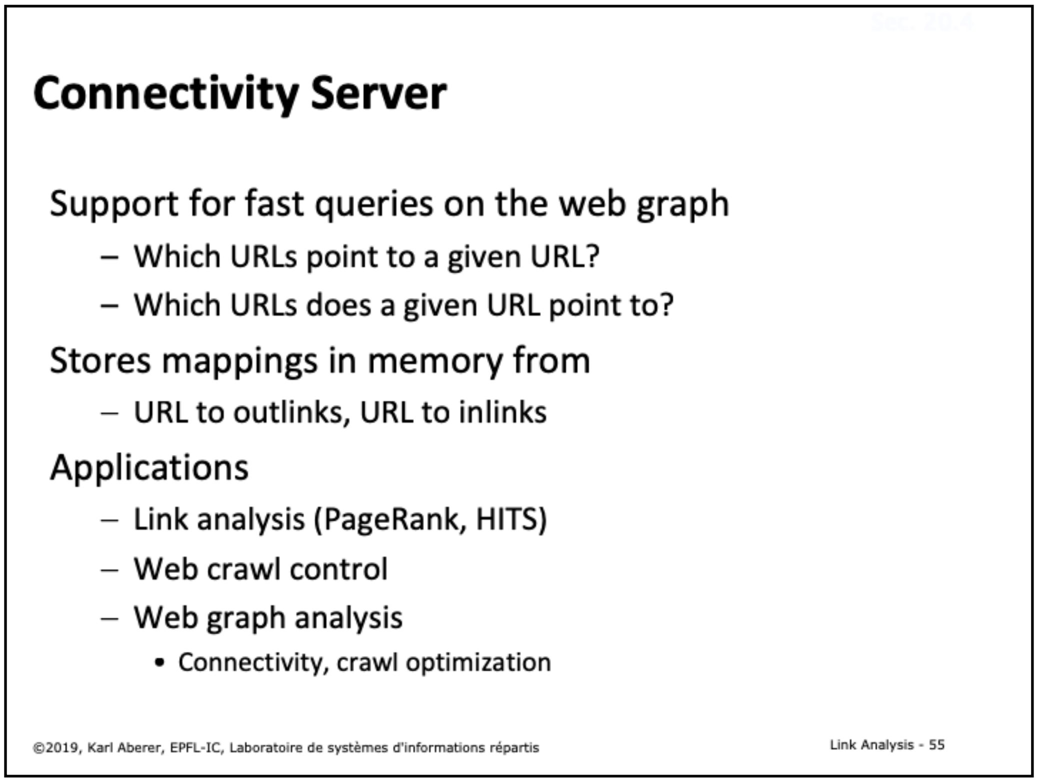
Implementation

- How to efficiently obtain the base set?

Similarly as for PageRank, the HITS algorithm is also vulnerable structural anomalies of the link structure, be it caused by conscious manipulation or by chance. Topic drift would imply that the set of base pages refers to a topic that is different from the original intended one, expressed by a search query. For example, if we would search for top European universities, it could well happen that the topic would drift to top US universities, when ranking hubs come into play that strongly refer to those. We could have also clusters of related pages, that in the iterative computation mutually boost each others scores, and thus give too high weight to pages that would not merit it. This problem is similar to the one of link spamming with link farms.

In terms of implementation, link-based ranking algorithms require the capability to efficiently retrieve the link graph for the Web. This problem is addressed by so-called connectivity servers that we will introduce next.

11. LINK INDEXING



Connectivity Server

Support for fast queries on the web graph

- Which URLs point to a given URL?
- Which URLs does a given URL point to?

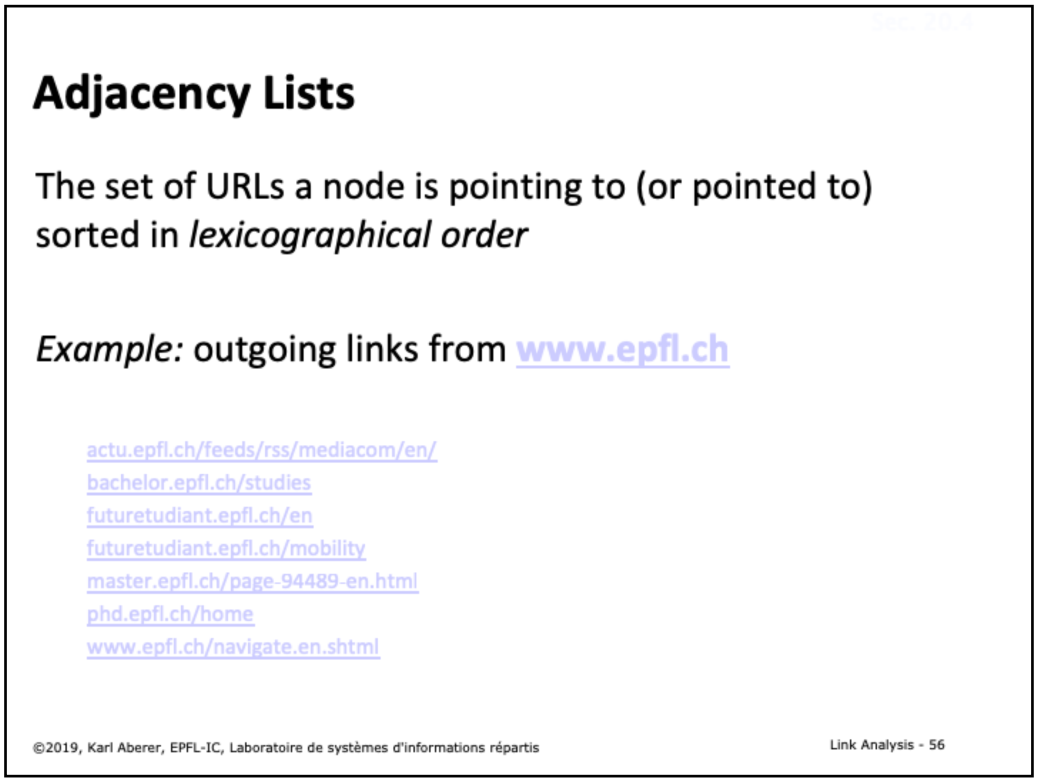
Stores mappings in memory from

- URL to outlinks, URL to inlinks

Applications

- Link analysis (PageRank, HITS)
- Web crawl control
- Web graph analysis
  - Connectivity, crawl optimization

In order to efficiently analyze the Web graph the Web links need to stored in a database, respectively index. A connectivity server is such an index. In essence, it stores for each URL all the URLs that the Web page at the URL is pointing to, and the URLs of other Web pages that point to this URL. Apart from link-based ranking algorithms as described before the applications are manifold.



Sec. 20.4

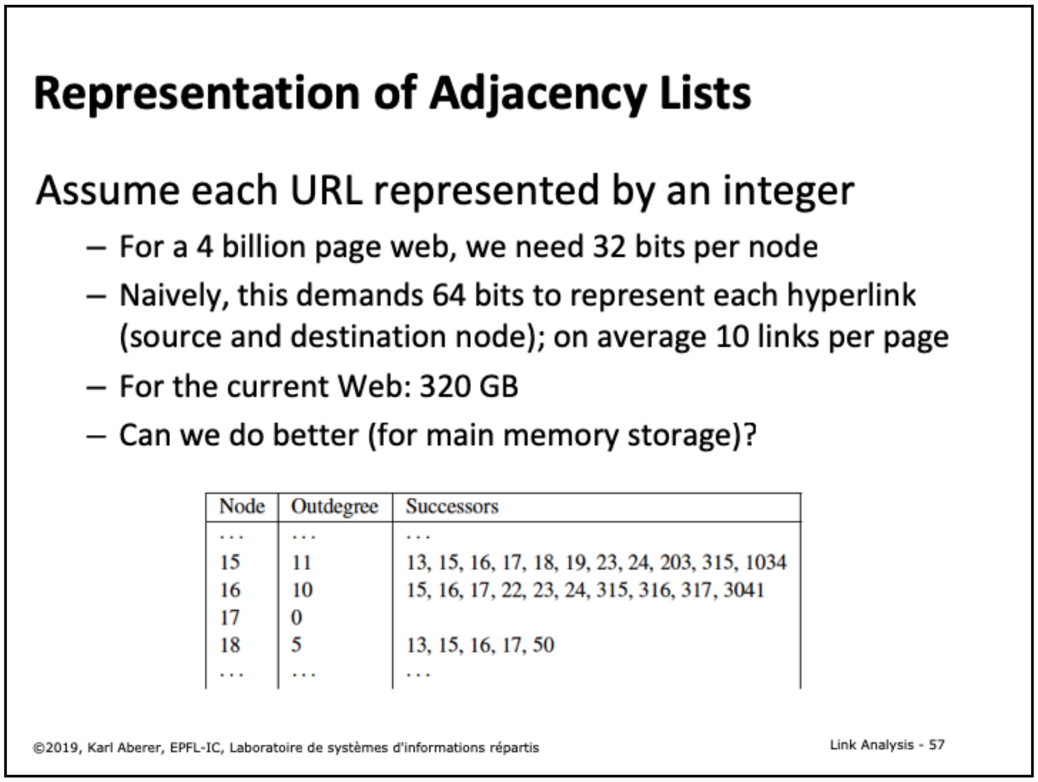
Adjacency Lists

The set of URLs a node is pointing to (or pointed to) sorted in *lexicographical order*

Example: outgoing links from www.epfl.ch

actu.epfl.ch/feeds/rss/mediacom/en/
bachelor.epfl.ch/studies
futuretudiant.epfl.ch/en
futuretudiant.epfl.ch/mobility
master.epfl.ch/page-94489-en.html
phd.epfl.ch/home
www.epfl.ch/navigate.en.shtml

A connectivity server has to store all outgoing (and incoming) links to a web page. For example, the home page of EPFL contains a large set of outgoing links, some of which are shown here. As a first step, the lists of links are sorted in lexicographical order. As a result we obtain the adjacency list, which is similar to the posting list of a document.



Representation of Adjacency Lists

Assume each URL represented by an integer

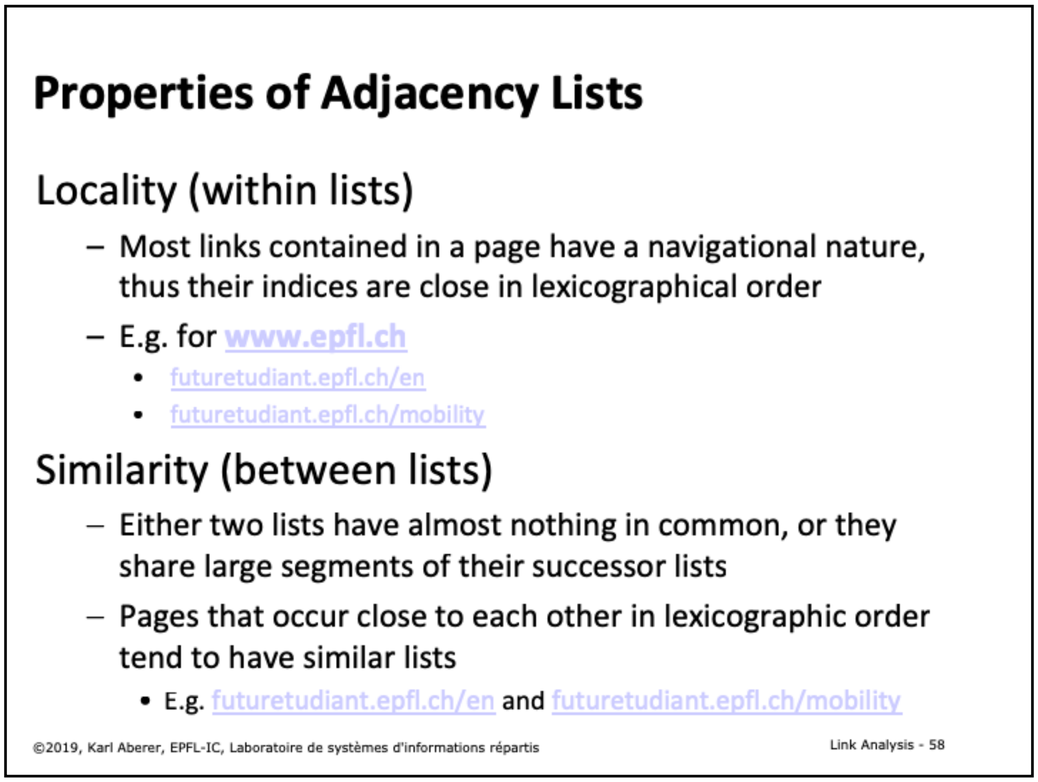
- For a 4 billion page web, we need 32 bits per node
- Naively, this demands 64 bits to represent each hyperlink (source and destination node); on average 10 links per page
- For the current Web: 320 GB
- Can we do better (for main memory storage)?

Node	Outdegree	Successors
...	...	...
15	11	13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	15, 16, 17, 22, 23, 24, 315, 316, 317, 3041
17	0	
18	5	13, 15, 16, 17, 50
...	...	...

As a first means to optimize the representation of adjacency lists, we represent each URL by one integer, instead of storing its textual form. From this we can estimate that for the current Web, we need 320 GB to represent all links:

- The current (crawled) Web has about 4 billion pages (http://www.worldwidewebsize.com/)
- It is estimated that a page contains on average 10 links
- We need 32 bits for each URL, which demands 64 bits for the storage of a single link.

Even with large memories this still a significant index size. In the following, we will show how to get this down to approximately ~3 bits/link which makes the index much more manageable. This will be achieved by systematically compressing the adjacency lists.


Properties of Adjacency Lists

Locality (within lists)

- Most links contained in a page have a navigational nature, thus their indices are close in lexicographical order
- E.g. for www.epfl.ch
  - futuretudiant.epfl.ch/en
  - futuretudiant.epfl.ch/mobility

Similarity (between lists)

- Either two lists have almost nothing in common, or they share large segments of their successor lists
- Pages that occur close to each other in lexicographic order tend to have similar lists
  - E.g. futuretudiant.epfl.ch/en and futuretudiant.epfl.ch/mobility

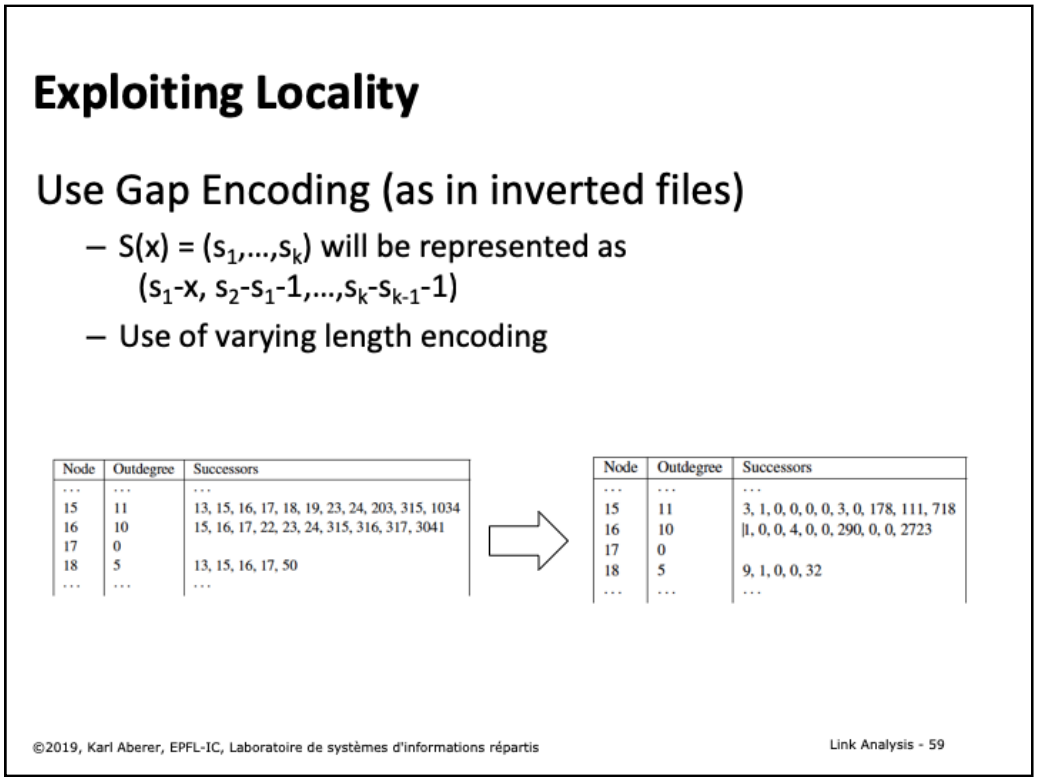
©2019, Karl Aberer, EPFL-IC, Laboratoire de systèmes d'informations répartis

Link Analysis - 58

For compressing adjacency lists we base ourselves on the following observations:

Locality. Most links contained in a page have a navigational nature: they lead the user to some other pages within the same host (as we can see clearly from the example of the EPFL home page). If we compare the source and target URLs of these links, we observe that they share often a long common prefix. Thus, if URLs are sorted lexicographically, the index of source and target are close to each other. Locality is a property of a list, thus addresses intra-list similarity.

Similarity. Pages that occur close to each other (in lexicographic order) tend to have many common successors. This is because many navigational links are the same within the same local cluster of pages, and even non-navigational links are often copied from one page to another within the same host. Similarity is a property concerning different lists, thus addresses inter-list similarity.



Exploiting Locality

Use Gap Encoding (as in inverted files)

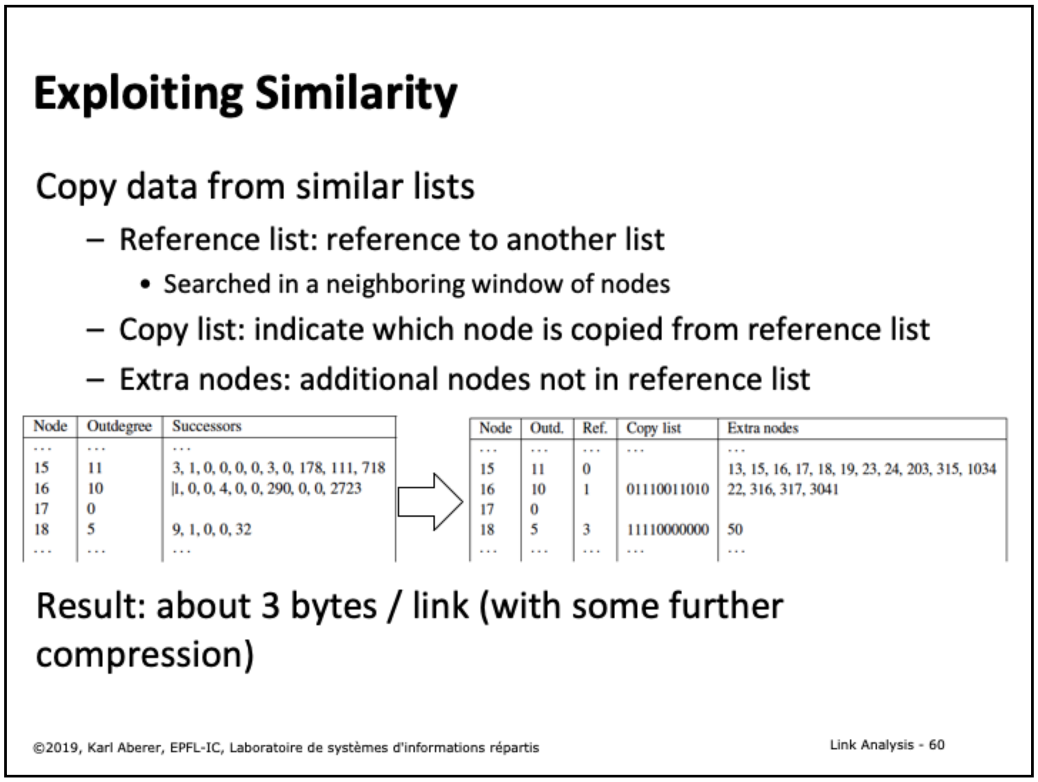
- $S(x) = (s_1,\dots,s_k)$ will be represented as $(s_1-x, s_2-s_1-1,\dots,s_k-s_{k-1}-1)$
- Use of varying length encoding

Node	Outdegree	Successors
...	...	...
15	11	13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	15, 16, 17, 22, 23, 24, 315, 316, 317, 3041
17	0	
18	5	13, 15, 16, 17, 50
...	...	...

$\Rightarrow$

Node	Outdegree	Successors
...	...	...
15	11	3, 1, 0, 0, 0, 0, 3, 0, 178, 111, 718
16	10	|1, 0, 0, 4, 0, 0, 290, 0, 0, 2723
17	0	
18	5	9, 1, 0, 0, 32
...	...	...

Locality can be exploited in a way analogous of how compression of posting lists for text indexing has been performed. Instead of storing the absolute values of the URL identifiers, differences are stored. In other words, we perform gap encoding. The resulting differences are then encoded using a varying length compression scheme, such as gamma coding.



Exploiting Similarity

Copy data from similar lists

- Reference list: reference to another list
  - Searched in a neighboring window of nodes
- Copy list: indicate which node is copied from reference list
- Extra nodes: additional nodes not in reference list

Node	Outdegree	Successors
...	...	...
15	11	3, 1, 0, 0, 0, 0, 3, 0, 178, 111, 718
16	10	|1, 0, 0, 4, 0, 0, 290, 0, 0, 2723
17	0	
18	5	9, 1, 0, 0, 32
...	...	...

Node	Outd.	Ref.	Copy list	Extra nodes
...	...	...	...	...
15	11	0		13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	1	01110011010	22, 316, 317, 3041
17	0			
18	5	3	11110000000	50
...	...	...	...	...

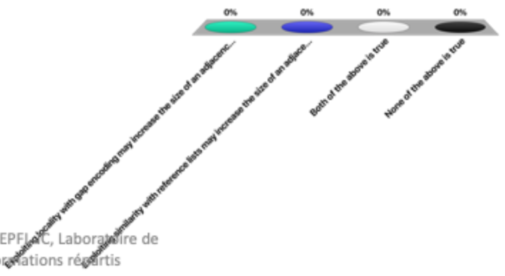
Result: about 3 bytes / link (with some further compression)

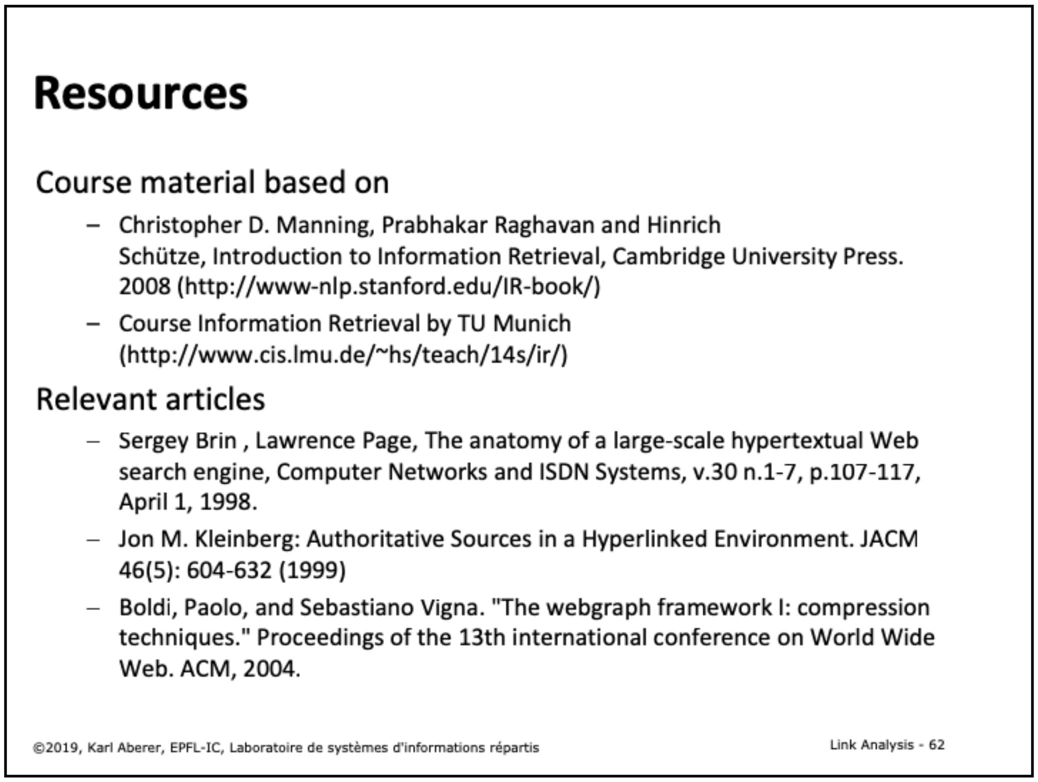
For exploiting similarity we apply a similar idea as gap encoding, but now applied to the differences among different adjacency lists. If we consider a reference list (in the example the list of Node 15), subsequent adjacency lists can story a bitlist that indicates which nodes of the reference lists are retained in the subsequent adjacency list, and which are not. 0 indicates that the node in the reference lists is not retained, and 1 indicates that it is retained. Since the subsequent adjacency list can also contain other nodes, that are not stored in the reference list, those extra nodes are stored explicitly.

Candidates for potential reference lists are searched among neighboring lists using a window of predefined size. The choice of the window size is critical, as larger windows increase chances of finding good candidates, but also increase the cost of compression

Together with some further compression applied to the copy lists and the extra nodes, this index compression achieves about 3 Bytes/link cost in the representation of the Web graph.

1. Exploiting locality with gap encoding may increase the size of an adjacency list
2. Exploiting similarity with reference lists may increase the size of an adjacency list
3. Both of the above is true
4. None of the above is true





Resources

Course material based on

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008 (http://www-nlp.stanford.edu/IR-book/)
- Course Information Retrieval by TU Munich (http://www.cis.lmu.de/~hs/teach/14s/ir/)

Relevant articles

- Sergey Brin , Lawrence Page, The anatomy of a large-scale hypertextual Web search engine, Computer Networks and ISDN Systems, v.30 n.1-7, p.107-117, April 1, 1998.
- Jon M. Kleinberg: Authoritative Sources in a Hyperlinked Environment. JACM 46(5): 604-632 (1999)
- Boldi, Paolo, and Sebastiano Vigna. "The webgraph framework I: compression techniques." Proceedings of the 13th international conference on World Wide Web. ACM, 2004.