Applications of Artificial Intelligence

– Winter Term 21/22 –

Chapter 03

Information Retrieval II

Prof. Dr. Adrian Ulges

RheinMain University of Applied Sciences

Outline

Benchmarking

Here: **Benchmarking** = measuring IR systems' **quality of results**.

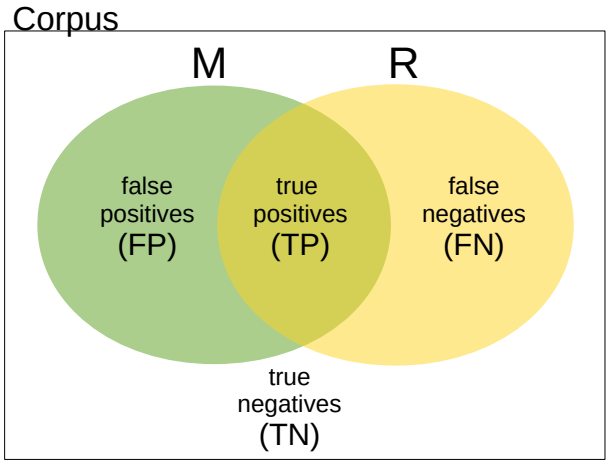
Why is Benchmarking important?

- controlling performance level
- **optimizing parameters**!
- **improving** IR systems

Example: A/B Testing in Web Search

- Company X develops a **new feature** for their scoring function *(e.g., PageRank)*.
- They redirect some queries to System B with the new features.
- They compare user satisfaction with the new features vs. without.

Benchmarking: Formalization



- Wir run a **test query q**
- The IR system returns a **subset M** of the corpus.
- In the corpus, there is another **subset R** of **relevant** documents.
- A **perfect result** would be: $M=R$.
- We divide the corpus into four subsets *(top right)*.

Are these Quality Measures "good"?

- error rate

$$\frac{\#FP + \#FN}{\#TP + \#TN + \#FP + \#FN}$$

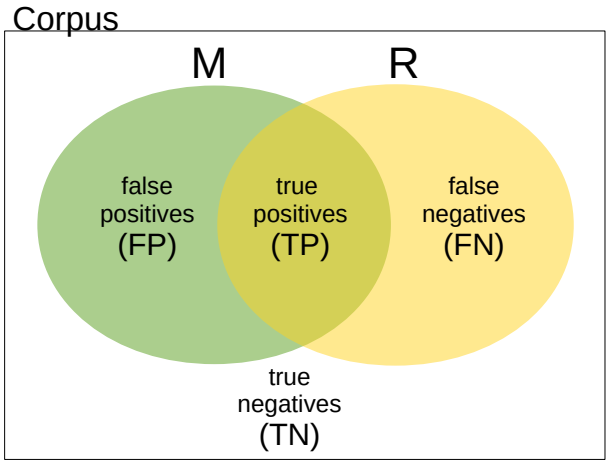
- false positive rate

$$\frac{\#FP}{\#FP + \#TN}$$

- false negative rate

$$\frac{\#FN}{\#FN + \#TP}$$

Benchmarking: Precision and Recall



These two might the **most important** quality measures in information retrieval:

Precision

$$\frac{\#(M \cap R)}{\#M}$$

Recall

$$\frac{\#(M \cap R)}{\#R}$$

Precision tells us...

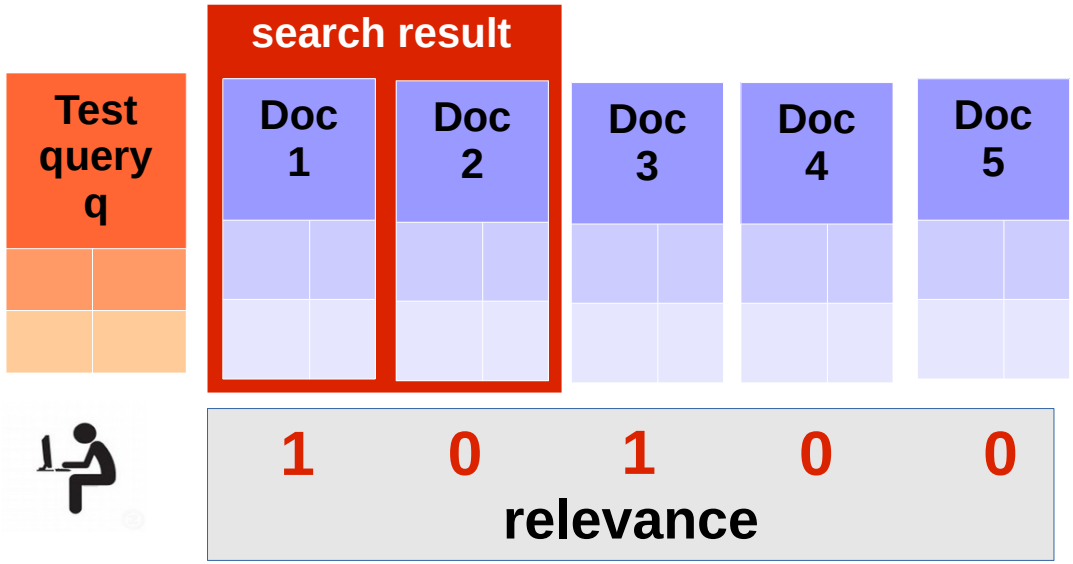
- what fraction of **retrieved documents** is **actually relevant**.
- precision is high if the system only returns a **few** top results.

Recall tells us...

- what fraction of **relevant documents** has **been retrieved**.
- The recall is usually high if the system returns **many** results.

Precision+Recall: Example

The IR system returns Doc 1, Doc 2



- true positives? Doc 1
- false positives? Doc 2
- true negatives? Doc 4,5
- false negatives? Doc 3

- false positive rate $= 1 / 3$
- false negative rate $= 1 / 2$
- precision $= 1 / 2$
- recall $= 1 / 2$

Evaluating Ranked Results

Limitation so far

- So far, we assume scoring to return a **(unordered) result set**.
- We would like our benchmarking to address the fact that "good" results should be at the **top** of the result list.

Approach 1

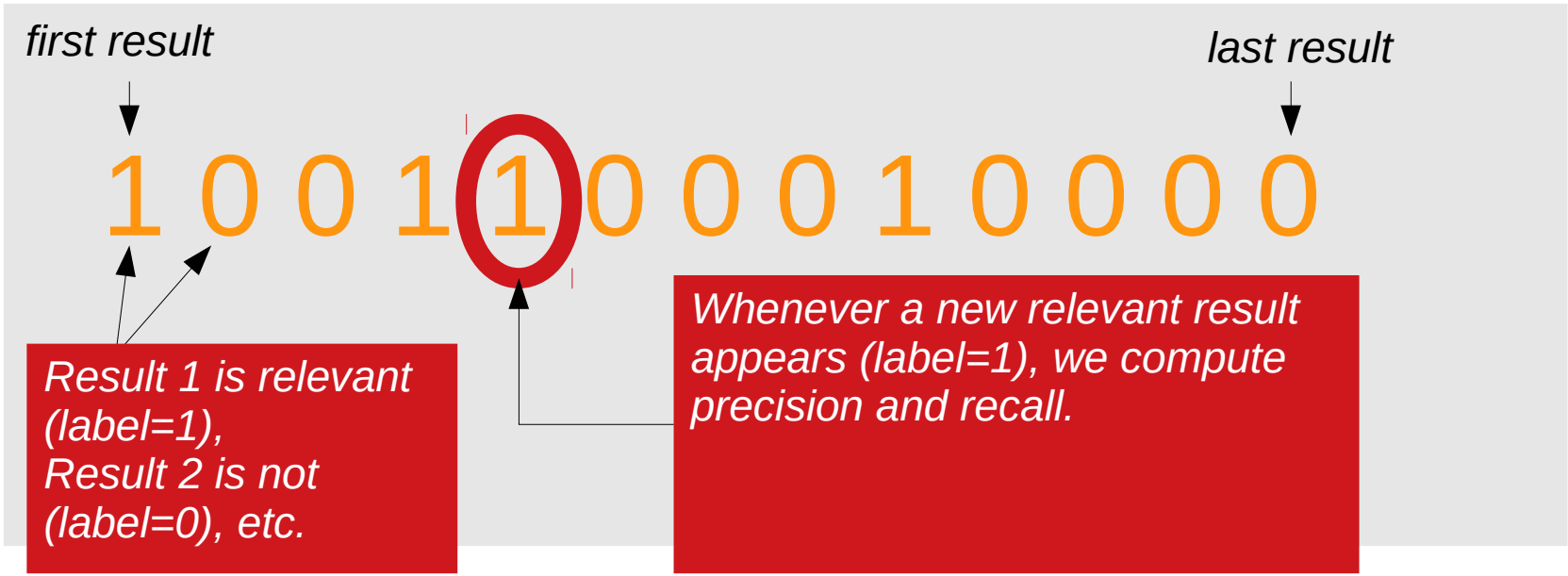
- We restrict the result list to the Top K results.
- → "Precision at Rank K" (short: **Prec@K**)
- Example (right): Prec@12 = 4/12.
- **Interpretation**: How many hits appear on the first result page?

Recall-Precision-Curves (RPCs)

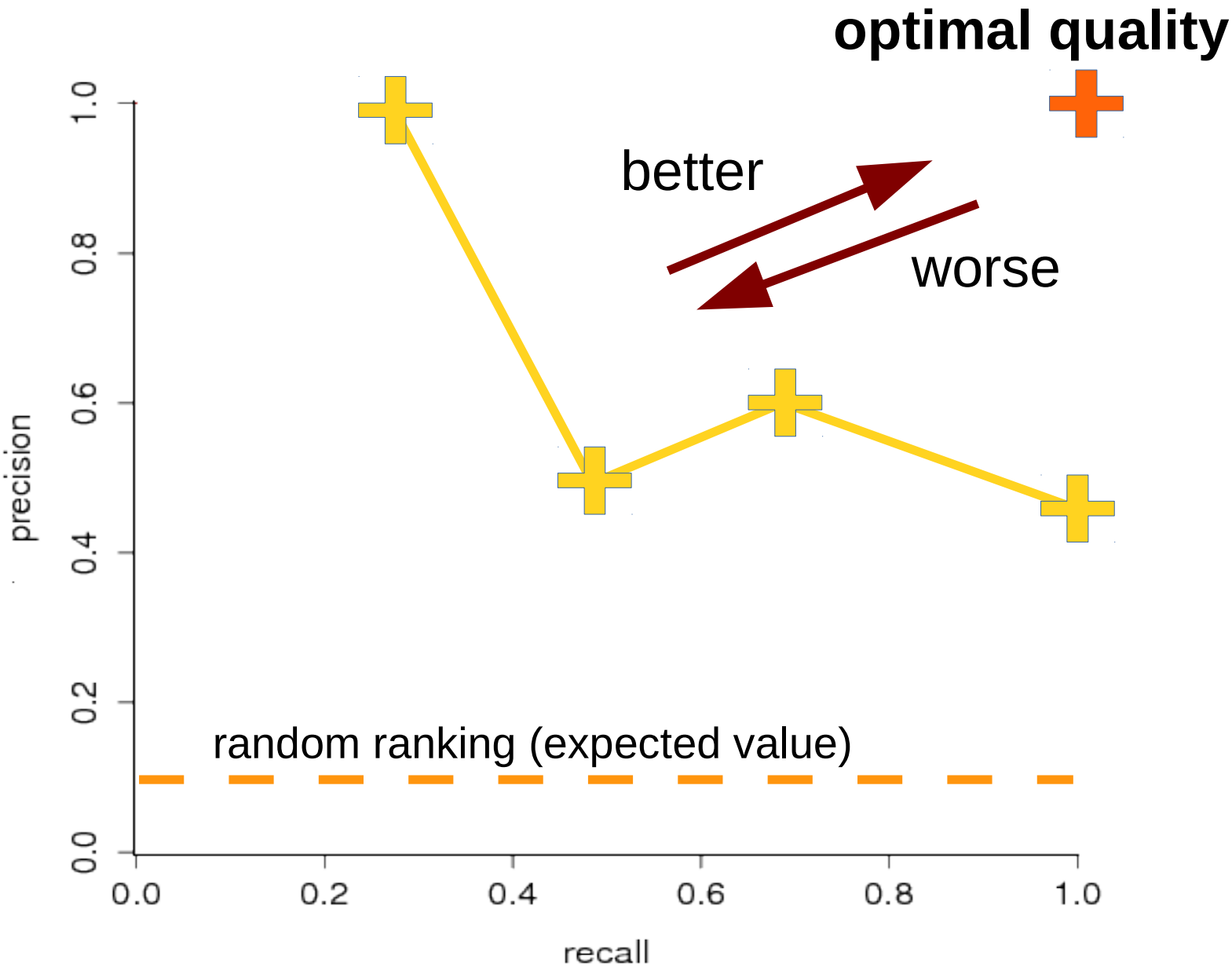
recall	prec.
0.25	1.00
0.50	0.50
0.75	0.60
1.00	0.44

Approach 2: Recall-Precision-Curves

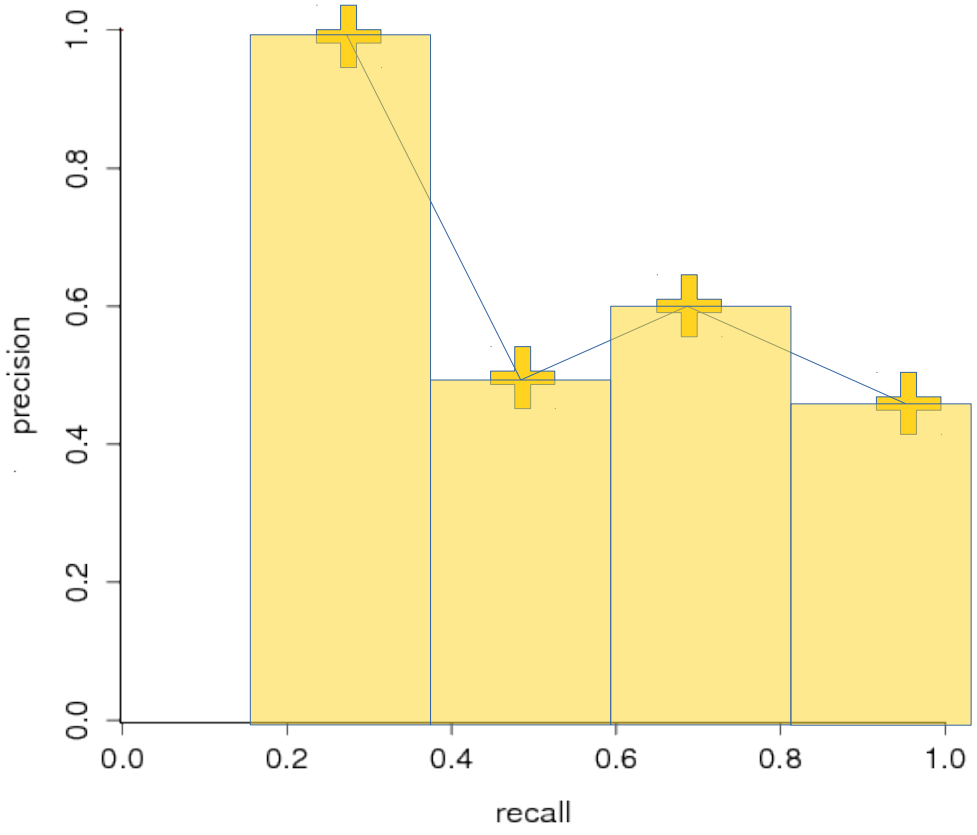
- We **vary** the length of the result list.
- Whenever a **new relevant item** is added, we measure precision and recall.
- With longer lists, precision will tend to decrease and recall will tend to increase.
- We obtain a curve, the **recall-precision curve (RPC)**.



RPCs: Example



RPCs: AP and MAP



- From the RPC curve, we derive a **performance indicator**: The **average precision (AP)** (= yellow area = average of all precision values in the RPC).
- By averaging over all test queries, we obtain the **mean average precision (MAP)**.

Benchmarking: Discussion

Does benchmarking – as introduced above – **truly reflect** search quality (as perceived by the **user**)?

- **binary relevance assessments (labels=0,1)** are inadequate *(example: user finds a solution by combining two particular documents)*.
- **Diversity of results is neglected** *(example: system returns* 100 *very identical documents)*
- Relevance assessments (*ground truth*) are always **subjective** *(solution: multiple labels for the same documents)*.
- Relevance assessments – when done by experts – are inherently **artificial**.
- **Visualization / UX** is not taken into account.
- ...

Outline

Benchmarking our Q&A Project

- Given: test questions with answers and passages.

Q : Who murdered Abraham Lincoln?
A : John Wilkes Booth.
P : John Wilkes Booth assassinated President Abraham Lincoln at Ford's Theatre in Washington, D.C., on April 14, 1865.

- In contrast to IR, there is only **one correct answer**.
- There should also be reward for approximate results *(e.g., John W. Booth)*!

Benchmarking our Q&A Project

Let $q_1, ..., q_n$ be the test questions with ground truth answers $a_1, ..., a_n$. For each question q_i, your QA system returns a ranked list of results $r_i^1, r_i^2, ..., r_i^K$. We define **result r_i^k's quality** as:

$$\mathcal{Q}(r_i^k) := \frac{1}{k} \times match(a_i, r_k^i)$$

- The **reciprocal rank 1/k** is highest if the result is ranked first, and decreases with the result's rank.
- The factor $match(a, r) \in [0, 1]$ determines the **textual similarity** between ground truth a and result r.
  For strings, it is the token Jaccard similarity:

$$match(a, r) := \begin{cases} \mathbf{1}_{a=r} & \text{if a is a number} \\ \frac{\#(a \cap r)}{\#(a \cup r)} & \text{else.} \end{cases}$$

Benchmarking our Q&A Project: Example

Q : Who murdered Abraham Lincoln?
A : John Wilkes Booth

Result list:

1. *John Wick*
2. *James Bond*
3. *John W. Booth*

Benchmarking our Q&A Project

We define a **result list's quality** as the best answer's quality ...

$$\mathcal{Q}(i) := max_k \mathcal{Q}(r_i^k)$$

... and the **overall quality** $\mathcal{Q}$ of your system as the total average:

$$\mathcal{Q} := \frac{1}{n} \cdot \sum_{i=1}^{n} \mathcal{Q}_i$$

Outline

Relevance Feedback: Motivation

Back to **IR Key Challenge (B) from Chapter 1**:
Due to the **semantic gap**, **choosing the "right" query** is hard.

Problem for QA: Low Recall

- For QA systems recall is more important than precision: answer processing can still figure out the best answer.
- Problem: We miss paraphrased statements

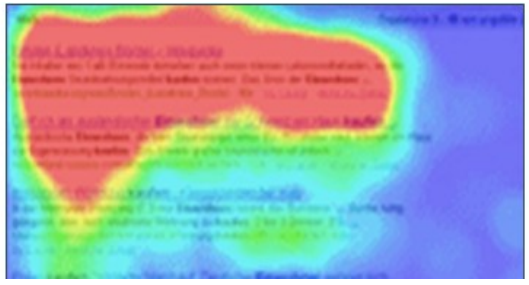
> **Question**: *"When did Trump speak to the media in Illinois?"*
> **Answer**: *"The President greeted the press in Chicago on Oct 27."*

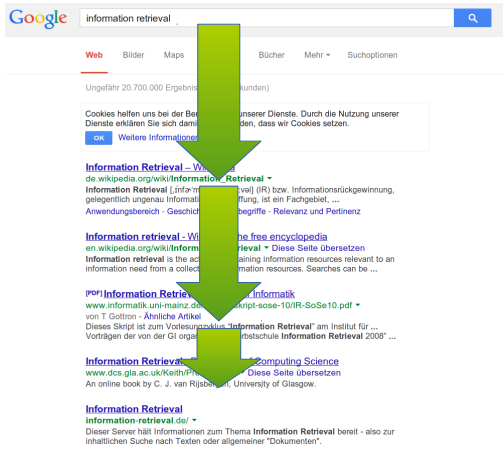
- **Approach** in the following: Improving the query based on **user feedback**.

Relevance Feedback: Pseudo Code

1. start with an initial query q_0.
2. set k:=0.
3. run a search with q_k.
4. the user labels some search results as relevant/irrelevant *(feedback F_k)*.
5. compute an improved query q_{k+1} based on $q_k, F_1, ..., F_k$.
6. set $k := k + 1$.
7. go to Step 3.

Feedback can be explicit or implicit





Rocchio's Algorithm

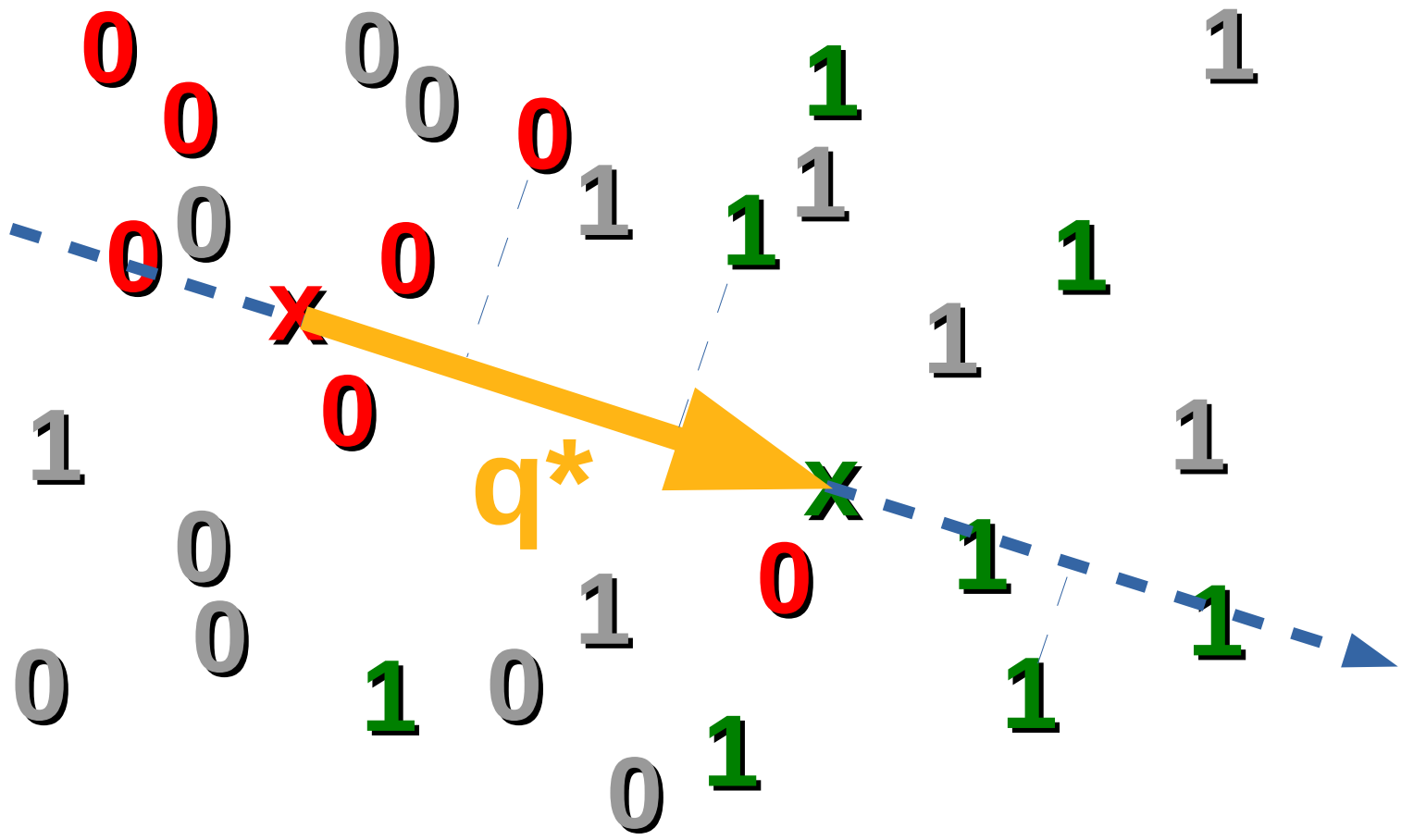
Question: How to compute q_{k+1} from q_k and feedback?

One Approach: Rocchio's Algorithm

- Queries und documents are vectors $\mathbf{q}$ / $\mathbf{d}$ *(cmp. vector space model)*.
- **Feedback** leads to two sets of documents: **relevant ones** (R^+) and **irrelevant ones** (R^-).
- We define

$$\mathbf{q}^{k+1} := \mathbf{q}_0 + \underbrace{\left(\frac{1}{\#R^+}\sum_{\mathbf{d}\in R^+}\mathbf{d}\right) - \left(\frac{1}{\#R^-}\sum_{\mathbf{d}\in R^-}\mathbf{d}\right)}_{q^*}$$

The Rocchio Modell: Illustration



The Rocchio-Modell: Example

Query: "refugees"

The user labels the documents ...

\+ `Syria refugee crisis: Facts you need to know`

\+ `number-syrian-refugees-passes-million`

\- `Australia's Immoral Refugees policy`

\+ `Syrian Refugees -- the latest from Al Jazeera`

\- `Climate Refugees: Climate Change and Migration`

Discussion

- terms such as `'syria'` or `'aleppo'` get **a weight** $> \mathbf{0}$.
- the query is **"expanded"** with these terms.
- Alternative: We simply add a few terms with highest (TF-IDF)-weight *(averaged over R^+)* to the query.

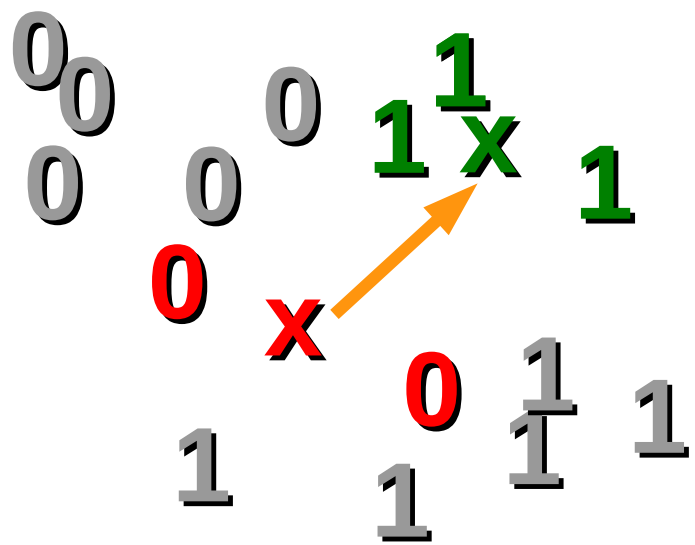
Das Rocchio-Modell (complete)

$$\mathbf{q}_{k+1} = \alpha \cdot \mathbf{q}_0 + \beta \cdot \left(\frac{1}{\#R^+} \sum_{\mathbf{d}^+ \in R^+} \mathbf{d}^+ \right) - \gamma \cdot \left(\frac{1}{\#R^-} \sum_{\mathbf{d}^- \in R^-} \mathbf{d}^- \right)$$

- $\alpha = 1, \beta = 0, \gamma = 0 \rightarrow q_{k+1} = q_0$ *(no relevance feedback)*
- Often, positive examples matter more than negative ones ($\beta > \gamma$). Default values: $\alpha = 1, \beta = 0.75, \gamma = 0.15$ [1]

Caution

- If the user labels only few documents, there can be mistakes.
- The model **overfits** to the small labeled document set.



Pseudo Relevance Feedback

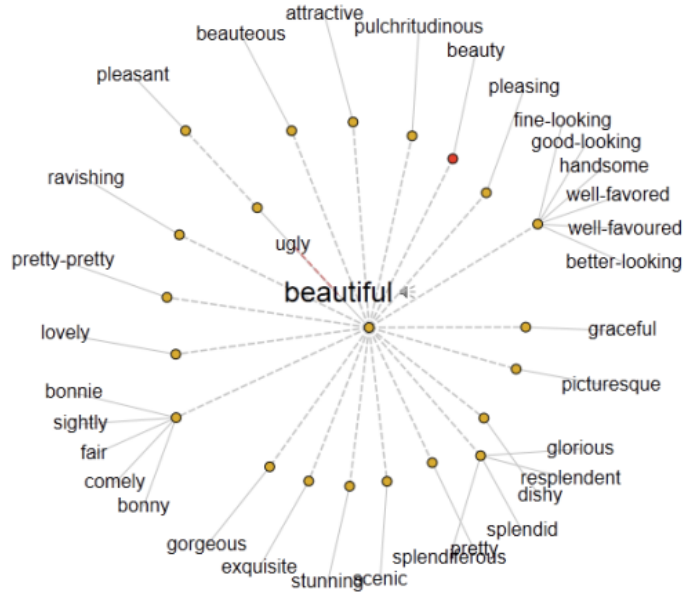
Problem: Relevance Feedback requires manual labeling

- This is not possible in our project...
- Web search: only 4% of users give feedback [1].

Approach: Pseudo Relevance Feedback

- simply assume that **top K documents** to be relevant.
- apply relevance feedback with these K documents.
- more risky and less accurate than true feedback ☹
- but: fully automatic! ☺

Query Expansion



Relevance feedback *(above)* adapts the query using the **result documents**.
Are there other strategies?

Query Expansion

- expand with words **similar** to the original query's words!

```
'laptop' → 'notebook'
```

Sources for "similar" words

- synonyms/homonyms/hypernyms from a thesaurus *(e.g. WordNet)*. **Problem**: thesauri are **static**.

```
'trump' → 'president'
```

- Statistical Learning of word similarities *(later)*.

Outline

PageRank

Idea

- Use the link structure between web documents to improve search results.
- **Links are indicators of the link target's authority!**

Examples

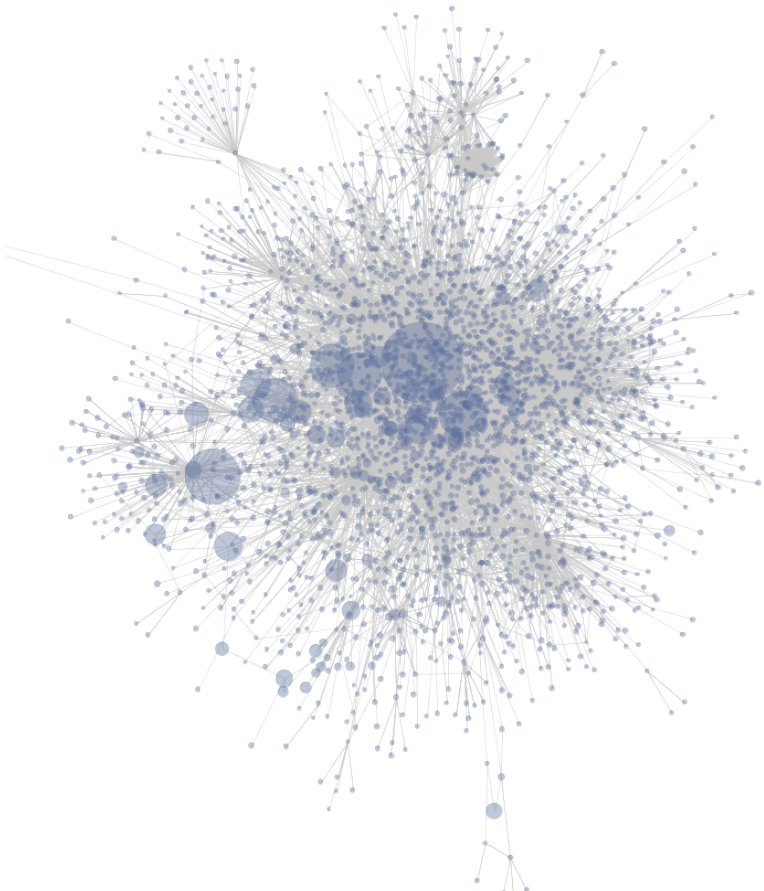
- Citations in science *(H-Index)*
- Number of followers on twitter

Frequently linked websites/documents

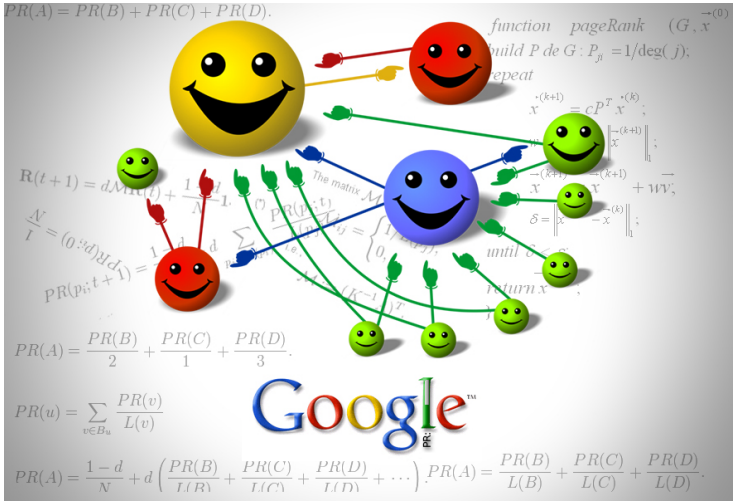
... have a "high authority"

... get higher scores

... are visited more frequently by crawlers.



PageRank: Introduction



- The **PageRank** [3] is a popular measure of authority.
- It is used as a boost factor for document scores in search engines.

Naive Approach

- authority := **number** of incoming links
- This approach is prone to **spam** *(bots adding links to manipulate scoring)*. ☹
- Links from **"important" pages** should **matter more**. ☹

Better Approach

- "Is a document linked by many important documents?"

The Random Surfer Model

Formalization

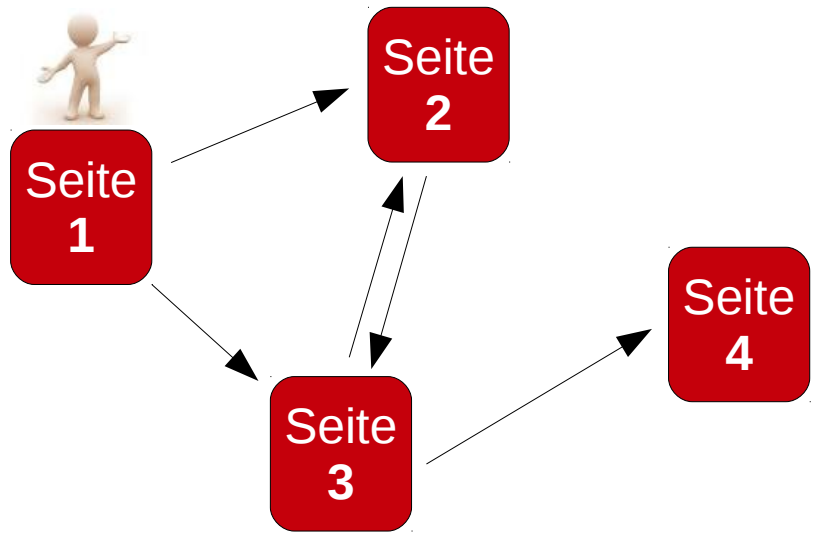
- Imagine a **user surfing randomly through the web**.
- The user starts at a **random page** p_0.
- The user **moves** from p_n to page p_{n+1}, by following one of p_n's outgoing links *(all links have the same probability)*.

Key Observation

- If a page is linked by other important pages, the surfer should visit the page **often**.
- **PageRank = distribution of the surfer's position!**

Random Surfer Model: Example

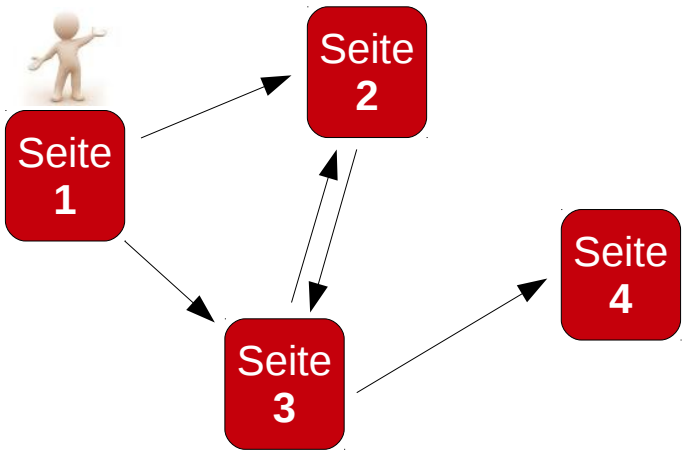
- The surfer starts at Page 1
- After 1 step
  - ... he is at Page 2 with 50% probability
  - ... he is at Page 3 with 50% probability.
- After 2 steps...?
- After 3 steps...?



Random Surfer Model: Formalization

- The random surfer visits a sequence of pages $S_1, S_2, S_3, \ldots,$.
- A **link** from Page j to Page i is a **tuple** (i, j)
- The **set of all links** $\mathbb{L}$ defines a transition matrix $T \in \mathbb{R}^{m \times m}$ (where m is the total number of pages):

$$T_{ij} = P(S_{n+1}{=}i \,|\, S_n{=}j) = \mathbf{1}_{(i,j)\in\mathbb{L}} \cdot \frac{1}{\#\{k \mid (k,j) \in \mathbb{L}\}}$$

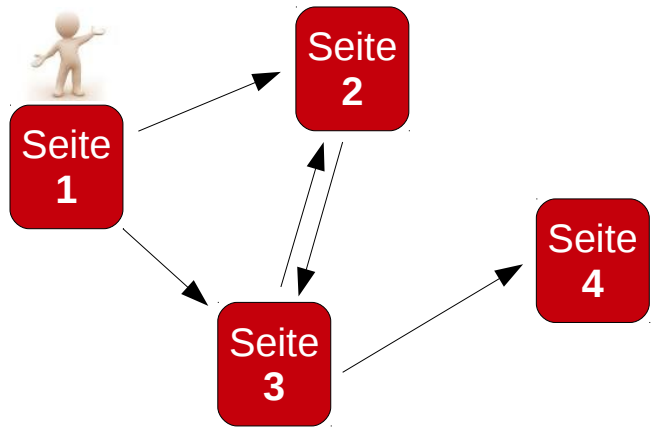


When on Page 3, the surfer moves with probability 0.5 to Page 2,

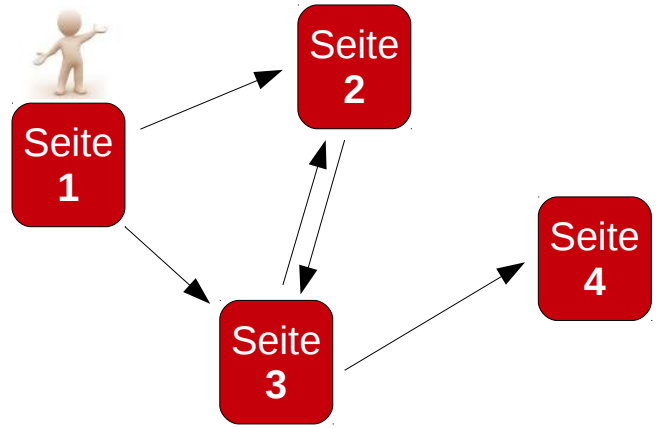
$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \end{pmatrix}$$

Random Surfer Model: Formalization

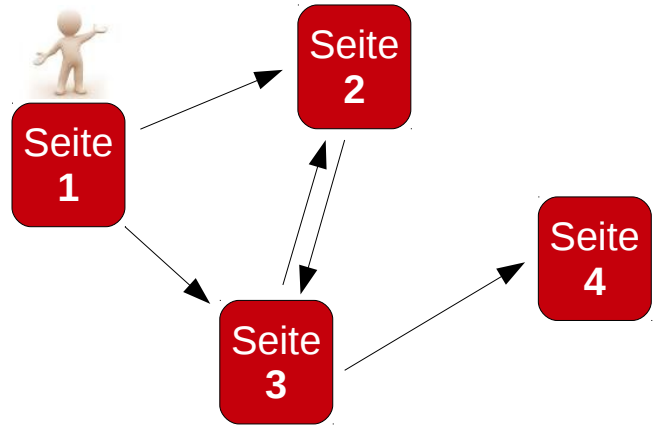
- After **n steps**, the surfer is located on each page with a certain probability.
- We collect these probabilities in a **vector $\mathbf{p}_n$** *(where is the surfer with which probability?)*



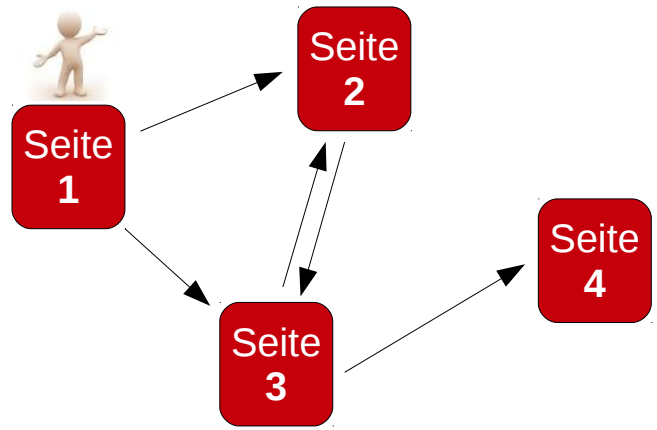
Random Surfer Model: Formalization



Random Surfer Model: Formalization



Random Surfer Model: Formalization



Random Surfer Model: Formalization

- **Problem**: **dead ends** (Page 4) → next state is undefined.
- **Trick**: **"teleporting"** → We allow equally distributed jumps to **any other** page.

1. **dead end**: jump randomly to any page.
2. **no dead end**: jump with probability α to any page.

Smoothed Transition Matrix T' *(k=total number of pages)*

$$T'_{ij} := \begin{cases} 1/m & \text{if Page } j \text{ is a dead end} \\ (1-\alpha) \cdot T_{ij} + \alpha \cdot 1/m & \textit{else}. \end{cases}$$

Example ($\alpha = 0.1$)

$$T = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0.5 & 1.0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{pmatrix} \quad T' = \begin{pmatrix} 0.025 & 0.025 & 0.025 & 0.25 \\ 0.475 & 0.025 & 0.475 & 0.25 \\ 0.475 & 0.925 & 0.025 & 0.25 \\ 0.025 & 0.025 & 0.475 & 0.25 \end{pmatrix}$$

Markov Chains

The surfer's mathematical model is called a **Markov chain**:

Definition (Markov Chain)

Let $S = \{1, 2, ..., m\}$ be a set of states. Let a random process generate a sequence of states $S_1, S_2, S_3, ... \in S$, such that for any time step $n \in \mathbb{N}$ holds:

$$P(S_{n+1}{=}s_{n+1} \,|\, S_1{=}s_1, ..., S_n{=}s_n) = P(S_{n+1}{=}s_{n+1} \,|\, S_n{=}s_n)$$

This process is a **discrete, finite, first-order Markov chain**.

Remarks

- The probabilities $P(S_{n+1}{=}j | S_n{=}i)$ correspond to the entries of our transition matrix T'_{ij}.

Markov Chains

Our surfer's particular Markov chain is

...irreducible:

- We can reach any state (i.e., any page) from any other one.

... aperiodic:

- For every state, there are at least two cycles of coprime length *(dt. teilerfremde Länge)*.

These properties...

- ... are both guaranteed through teleporting *(why?)*.

Markov-Ketten: Scrabbling

Random Surfer: Convergence

For **irreducible, aperiodic Markov chains**, the following holds:

- The surfer's distribution $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots$ converges to a **limit** $\mathbf{p}$.
- "If the surfer **surfs infinitely long**, we obtain a fixed **distribution of his state** over the pages".
- We also call $\mathbf{p}$ the **stationary distribution** of T'.

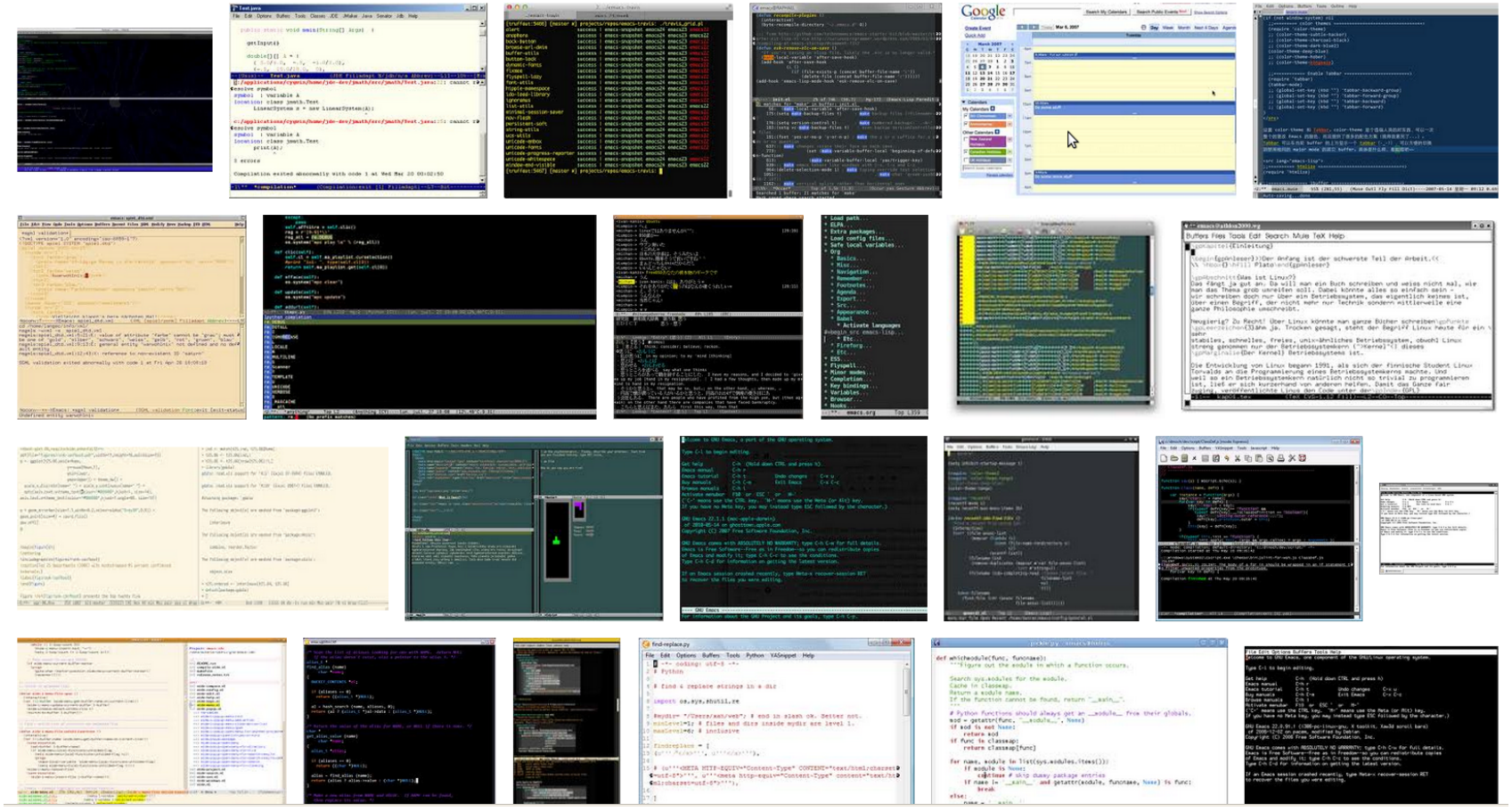
How to compute $\mathbf{p}$?

- $\mathbf{p}$ should not change with a single step, i.e.:

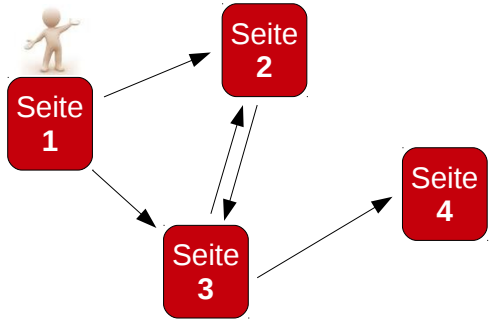
$$T \cdot \mathbf{p} = \mathbf{p}$$

- We see: $\mathbf{p}$ is an **eigenvector** of T!

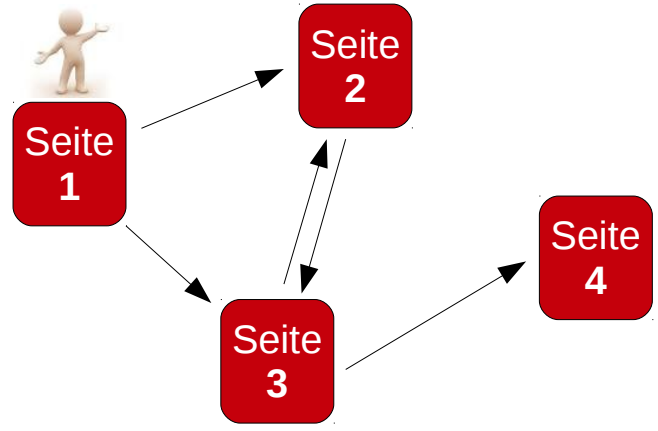
Random Surfer: Example



$$T = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0.5 & 1.0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{pmatrix}$$



Random Surfer → PageRank



A page's "PageRank" ...

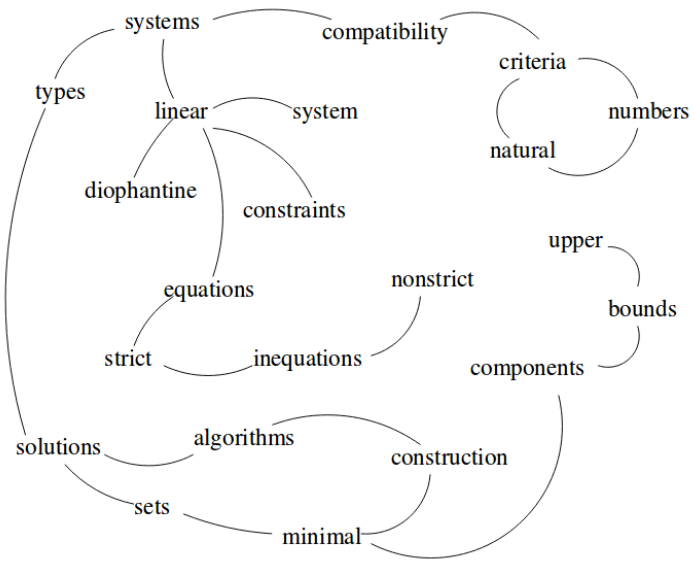
- ... ist just the corresponding entry in $\mathbf{p}$.
- Wer compute $\mathbf{p}$ by starting from distribution $\mathbf{p}_1$ and multiplying with T again and again:

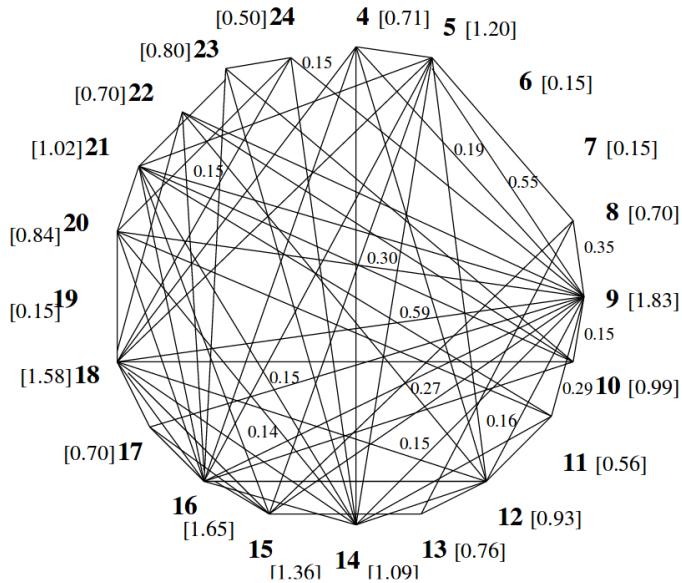
$$\mathbf{p} = \ldots \cdot T \cdot T \cdot T \cdot \mathbf{p}_1 = \lim_{k \to \infty} T^k \cdot \mathbf{p}_1 \approx T^{100} \cdot \mathbf{p}_1 = \begin{pmatrix} 0.082 \\ 0.289 \\ 0.378 \\ 0.252 \end{pmatrix}$$

Interpretation?

- Page 1 has no links → lowest PageRank.
- Page 3 has the most valuable links.
- Page 2 has (compared with Page 4) one more link, and thus a slightly higher PageRank.

Outlook: TextRank





- We can analyze **general link structures** with PageRank!
- **Example**: states = **words** or **sentences** of a text.
- **words** *(left)* are linked when close to each other.
- **sentences** *(right)* are linked when containing *similar words*.
- PageRank (here: *TextRank* [2]) generates **key terms** und **summaries** of texts.

References I

[1] C. Manning, P. Raghavan, and H. Schütze.
Introduction to Information Retrieval.
Cambridge University Press, 2008.

[2] Rada Mihalcea and Paul Tarau.
Textrank: Bringing order into texts.
In Proc. EMNLP 2004, pages 404–411, Barcelona, Spain, July 2004.

[3] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd.
The pagerank citation ranking: Bringing order to the web.
Technical Report 1999-66, Stanford InfoLab, 1999.