Anwendungen der KI

– Sommersemester 2018 –

Kapitel 03: Information Retrieval (Teil II)

Prof. Dr. Adrian Ulges
B.Sc. Informatik (AI, ITS, MI, WI)
Fachbereich DCSM
Hochschule RheinMain

Outline

Benchmarking

Benchmarking = Beurteilung der **Güte der Ergebnisse** eines IR-Systems

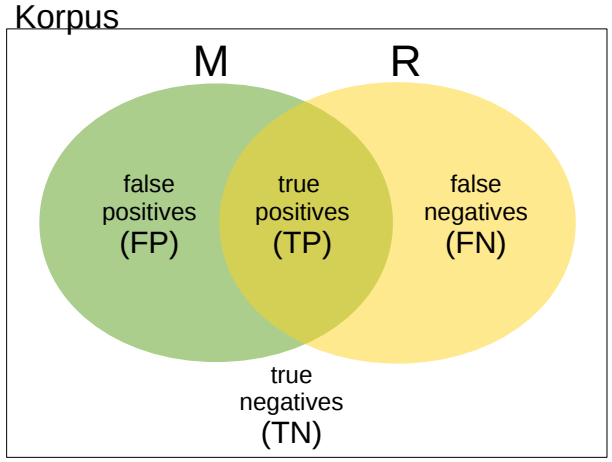
Warum ist Benchmarking wichtig?

- Betriebswirtschaftliche Steuerung
- Grundlage der **Optimierung von Parametern**!
- **Weiterentwicklung** von IR-Systemen

Praxisbeispiel: Web-Suchmaschine

- Entwicklung eines neuen **Features** für die Scoring-Funktion *(z.B. PageRank)*
- Umleitung eines Anteils der Queries auf ein Zweitsystem
- Vergleichendes Benchmarking **(A/B-Testing)**

Benchmarking: Formalisierung



- Wir stellen einen **Test-Query q**
- Das System liefert eine **Teilmenge M** des Korpus zurück.
- Im Korpus existiert eine **Teilmenge R** von **relevanten** Dokumenten R.
- Ein **perfektes Ergebnis** wäre: $M = R$.
- Wir unterteilen den Korpus in vier Teilmengen *(rechts oben)*.

Gütemaße

- Fehlerrate

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

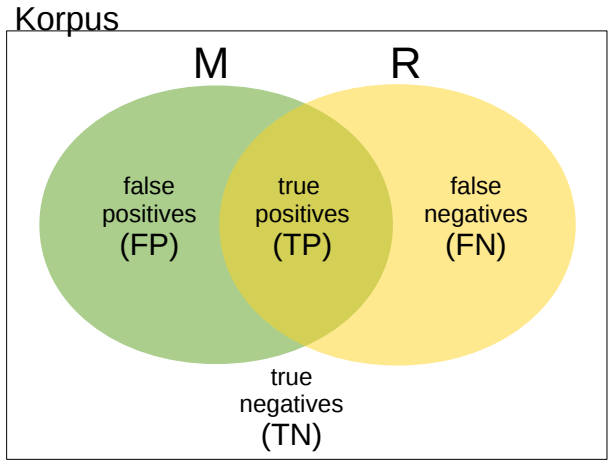
- False-Positive-Rate

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

- False-Negative-Rate

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

Benchmarking: Precision und Recall



Am **häufigsten** werden im Information Retrieval die folgenden beiden Maße verwendet:

Precision

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

Recall

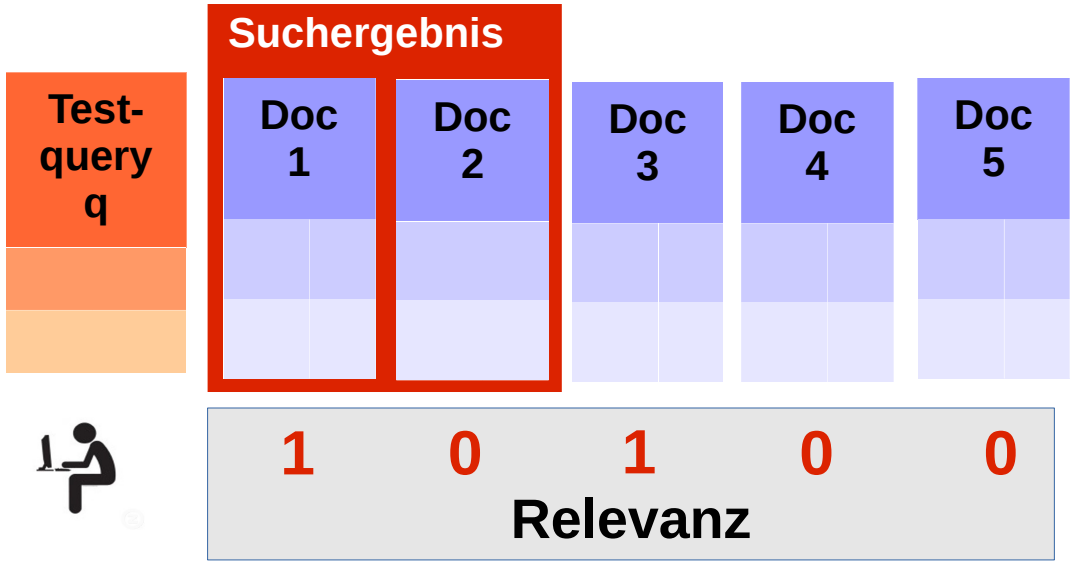
$$\frac{\#(M \cap R)}{\#R}$$

Anmerkungen

- Die **Precision** verrät uns, welcher Prozentsatz der **gefundenen Dokumente** auch **wirklich relevant** ist.
- Der **Recall** verrät uns, welcher Prozentsatz der **relevanten Dokumente** auch **wirklich gefunden** wurde.

Precision und Recall: Beispiel

Das IR-System liefert Doc 1, Doc 2



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

- Fehlerrate = 2 / 5
- False-Positive-Rate = 1 / 3
- False-Negative-Rate = 1 / 2
- Recall = 1 / 2
- Precision = 1 / 2

Score-basierte Evaluation

Einschränkung bisher

- IR-Systeme liefern nicht nur eine Ergebnismenge, sondern sie **ranken** Ergebnisse anhand eines Scores.
- Das Benchmarking sollte dies berücksichtigen: **Wo** in der Ergebnisliste taucht ein relevantes Dokument auf?

Ansatz 1

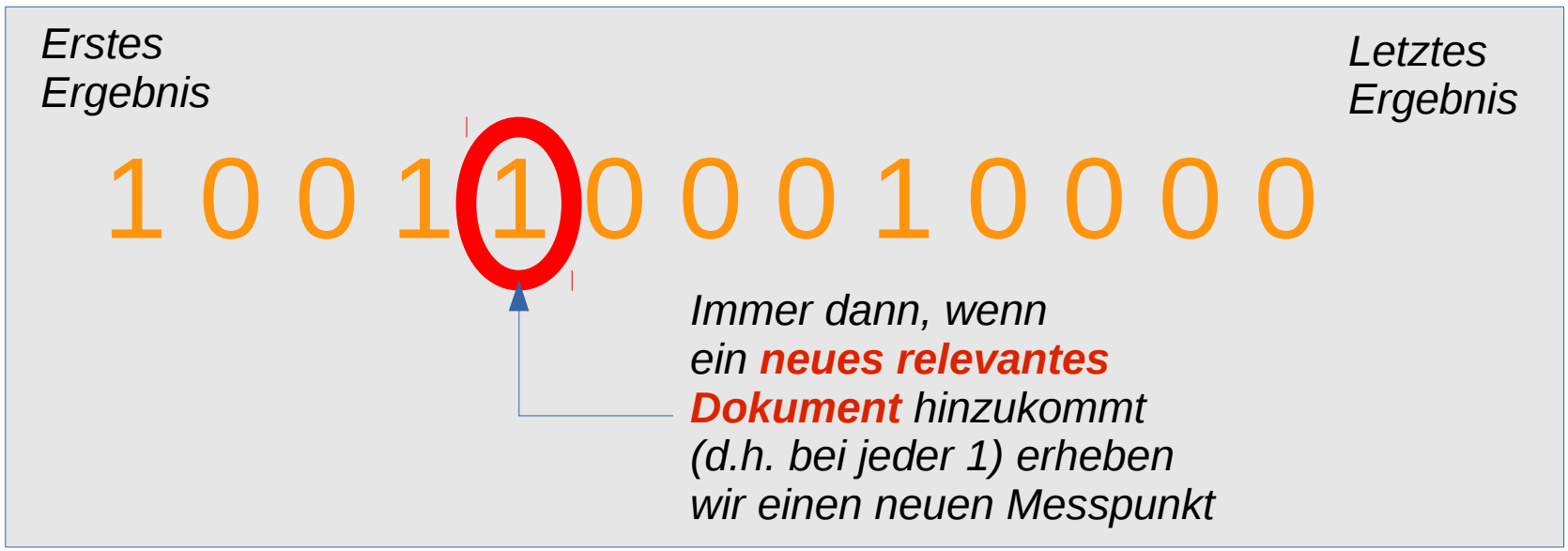
- Schneide die Ergebnisliste an **Position K** ab.
- **Standardmaß**: "Precision at Rank K" (z.B. **Prec@10**)
- **Interpretation**: Wieviele Treffer befinden sich z.B. auf der ersten Ergebnisseite?

Recall-Precision-Curves (RPCs)

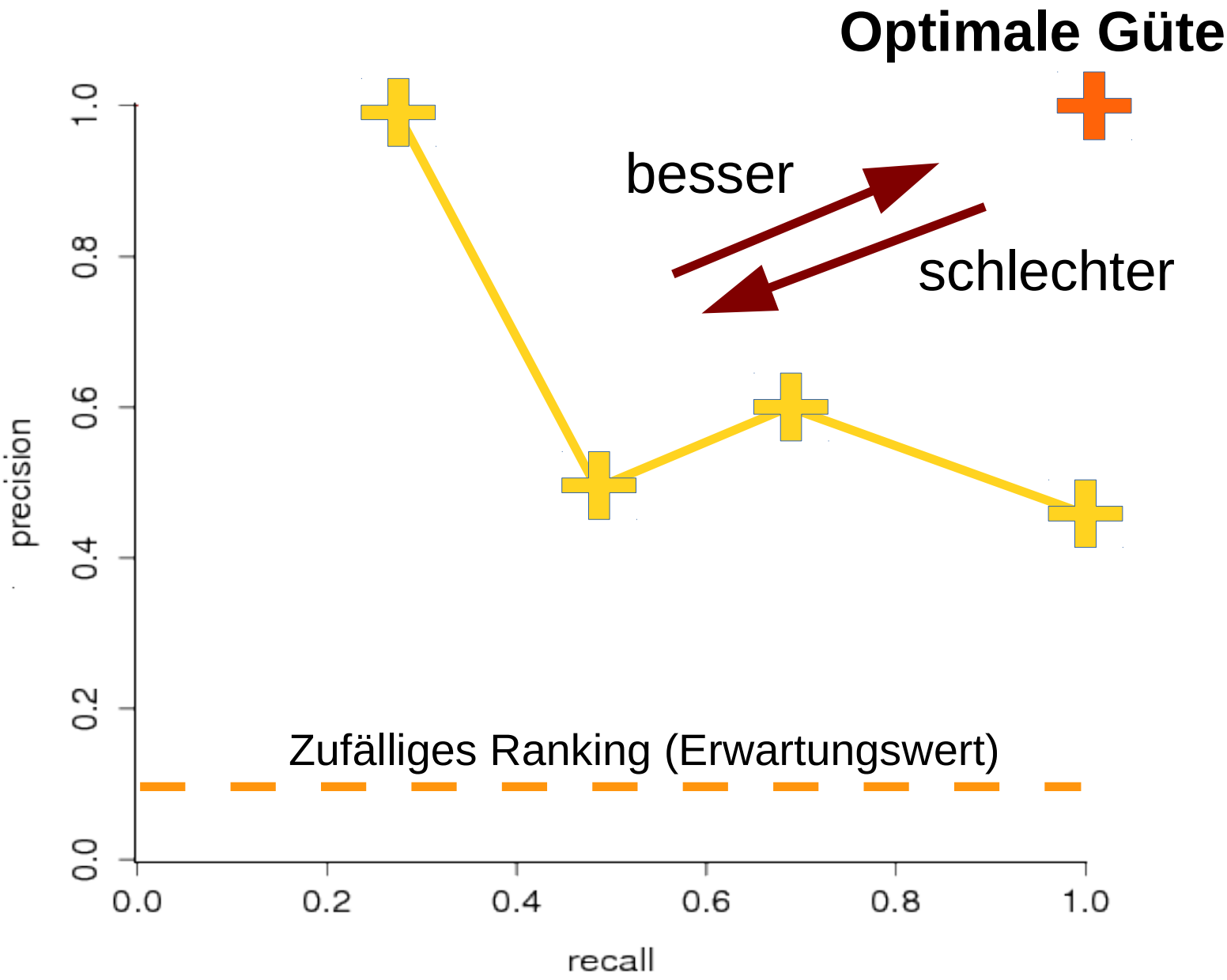
recall	prec.
0.25	1.00
0.50	0.50
0.75	0.60
1.00	0.44

Ansatz 2: RPC

- Lassen wir das IR-System **mehr und mehr** Ergebnisse liefern, **erhöht** sich der **Recall**. Die **Precision verringert** sich tendenziell.
- Wir **variieren** die **Länge der Ergebnisliste** und messen jeweils Precision und Recall.
- Es entsteht eine **Kurve**, die sogenannte **Recall-Precision-Curve (RPC)**.

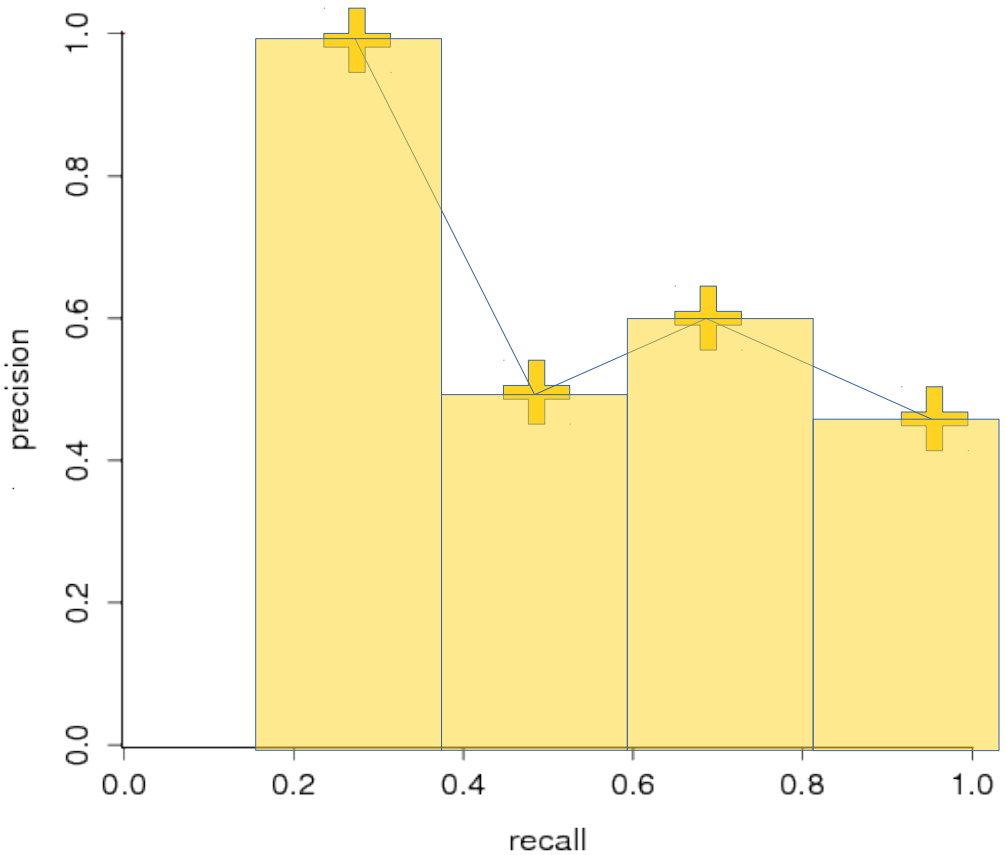


RPCs: Beispiel



RPCs: AP und MAP

- Aus der RPC-Kurve leiten wir einen **globalen Indikator** ab: Die **Average Precision (AP)** (= markierte Fläche = Mittlung aller Precision-Werte der RPC).
- Durch Mittlung über alle Test-Queries erhalten wir die **Mean Average Precision (MAP)**.



Benchmarking: Diskussion

Was sind die **Einschränkungen** des hier vorgestellten Benchmarking-Ansatzes?

- **Binäre Relevanz** trifft oft nicht zu
  (Beispiel: Benutzer findet die Lösung seines Problems durch Kombination mehrerer Zieldokumente).
- Die Labels (*Ground Truth*) ist evtl. **subjektiv**.
- Die Labels (*Ground Truth*) ist **künstlich**.
- **Präsentation** der Ergebnisse / UX nicht mit berücksichtigt.
- Die **Diversität** der Ergebnisse ist nicht berücksichtigt.
- ...

Benchmarking: Standard-Datensätze

TABLE 4.3 Common Test Corpora

Collection	*NDocs*	*NQrys*	*Size (MB)*	*Term/Doc*	*Q-D RelAss*
ADI	82	35			
AIT	2109	14	2	400	>10,000
CACM	3204	64	2	24.5	
CISI	1460	112	2	46.5	
Cranfield	1400	225	2	53.1	
LISA	5872	35	3		
Medline	1033	30	1		
NPL	11,429	93	3		
OSHMED	34,8566	106	400	250	16,140
Reuters	21,578	672	28	131	
TREC	740,000	200	2000	89-3543	x 100,000

Benchmarking: Daten-Akquise

- Noch eine Einschränkung: **Aufwand zur Annotation**
- **Beispiel**: 1 mio. Dokumente, 2000 Queries, 2,5 Sekunden pro Label, 10$ pro Stunde
  → 14 Mio. $ Gesamtkosten für Annotation
- Wie kommen wir mit **weniger Labels** aus?

Beliebter Ansatz: Pooling

- Wir wollen mehrere Systeme **vergleichen**.
- Wir fassen die **die Top-N-Ergebnisse** der verschiedenen Systeme zum sogenannten **Pool** zusammen.
- Wir annotieren **nur den Pool**.
- Wir nehmen an, dass alle Dokumente außerhalb des Pools nicht-relevant sind.

Outline

Relevance Feedback: Motivation

Zurück zu unseren zwei **Schlüssel-Herausforderungen** im Information Retrieval

1. Die Unstrukturiertheit der Zieldaten
2. Die Wahl des "richtigen" Queries

Häufiges Problem: Geringer Recall

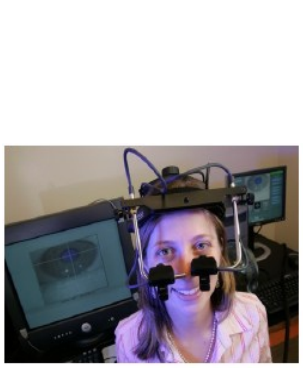
> **Query**: *"Trump speaks to the media in Illinois"*
> **Dokument**: *"The President greets the press in Chicago."*

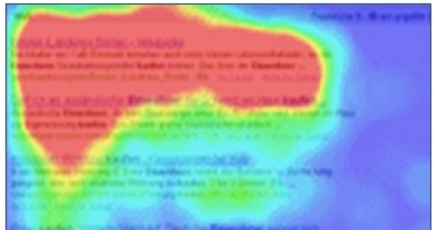
- Um den Recall zu erhöhen, versucht der Benutzer seine Anfrage **iterativ zu verbessern**.
- Die "richtigen" Query-Terme zu finden ist **schwer**.
- **Aber**: Ein **Treffer-Dokument** zu bewerten *(relevant/irrelevant)* ist **leicht**!

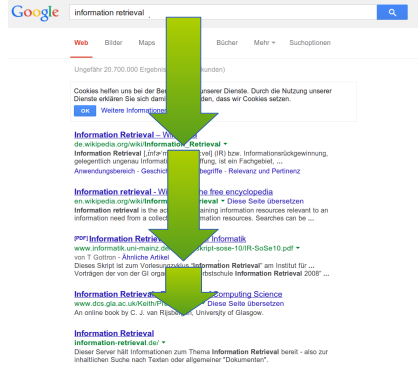
Relevance Feedback: Pseudo-Code

1. Der Benutzer formuliert eine initiale Anfrage q_0.
2. Setze $k := 0$.
3. Führe eine Suche mit q_k durch.
4. Der Benutzer bewertet einige Suchergebnisse als relevant/irrelevant *(Feedback)*.
5. Eine verbesserte Anfrage q_{k+1} wird automatisch berechnet.
6. Setze $k := k + 1$.
7. Gehe zu Schritt 3.

Feedback kann explizit oder implizit sein







Das Rocchio-Modell

Schlüsselfrage: Wie berechnen wir q_{k+1} aus q_k und Feedback?

- **Zwei Ansätze**: (1) **Rocchio**, (2) **probabilistisch** *(später)*.

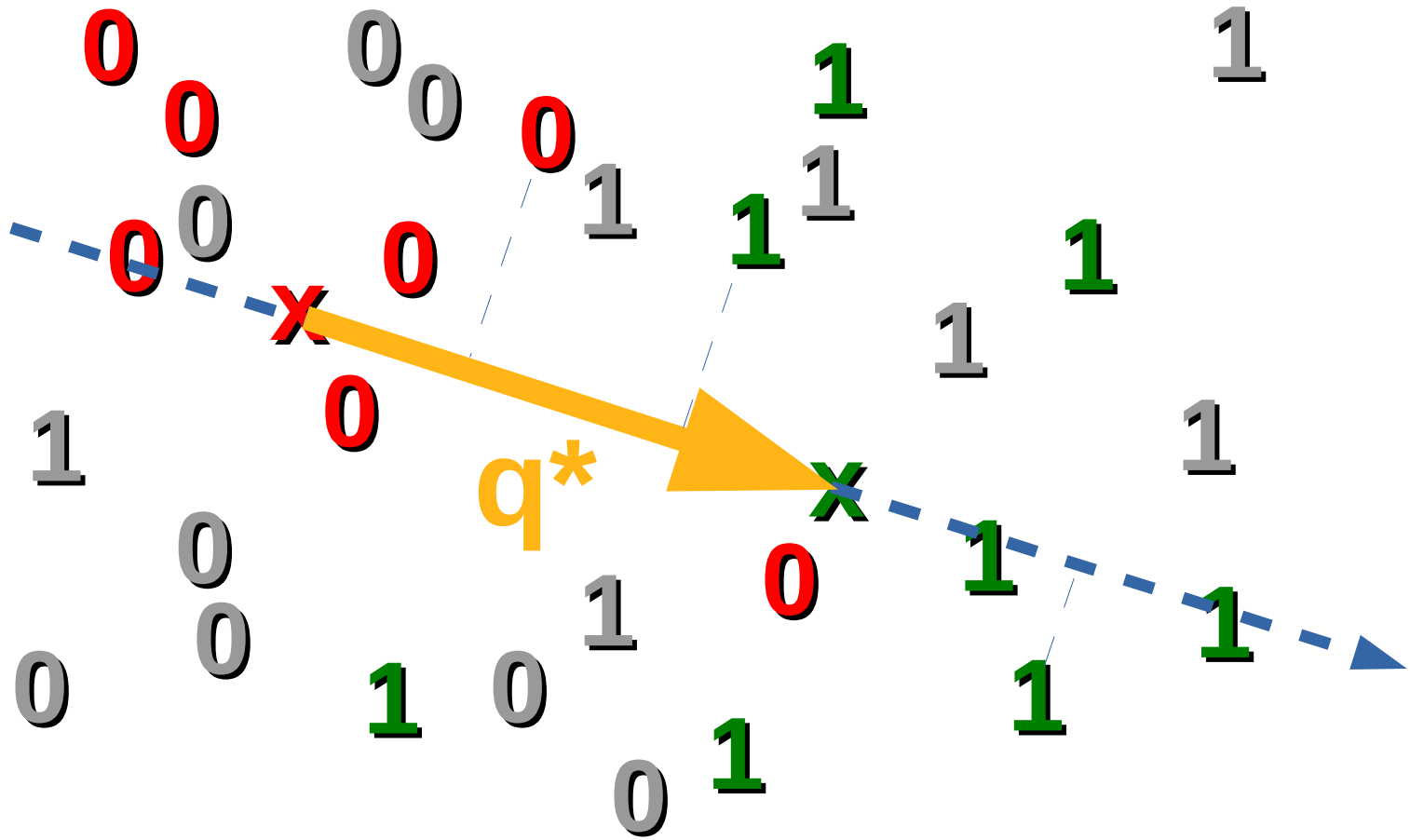
Relevance Feedback nach Rocchio

- Basiert auf dem **Vektorraum-Modell**: Queries und Dokumente sind (Bag-of-Words-)Vektoren $\mathbf{q}$ / $\mathbf{d}$.
- Das **Feedback** des Benutzers resultiert in zwei (kleinen) **Mengen**: Als **relevant/irrelevant markierte** Dokumente R^+ und R^-.
- Wir berechnen einen Hilfsvektor $\mathbf{q}^*$ ...

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

- ... und wählen *(im einfachsten Fall)*: $\mathbf{q}_{k+1} := \mathbf{q}_0 + \mathbf{q}^*$

Das Rocchio-Modell: Illustration



Das Rocchio-Modell: Erläuterung

Query: "refugees"

Der User bewertet die Dokumente...

\+ `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`

Was ist der Effekt von Relevance Feedback?

- Zusätzliche Terme wie `'syria'` oder `'aleppo'` bekommen im Query-Vektor **q** auf nun **ein Gewicht** $> \mathbf{0}$.
- Der Query wird **"expandiert"**.
- Alternativ können wir dem Query einfach einige **Top-Terme** mit hohem (TF-IDF)-Gewicht *(gemittelt über R^+)* hinzufügen.

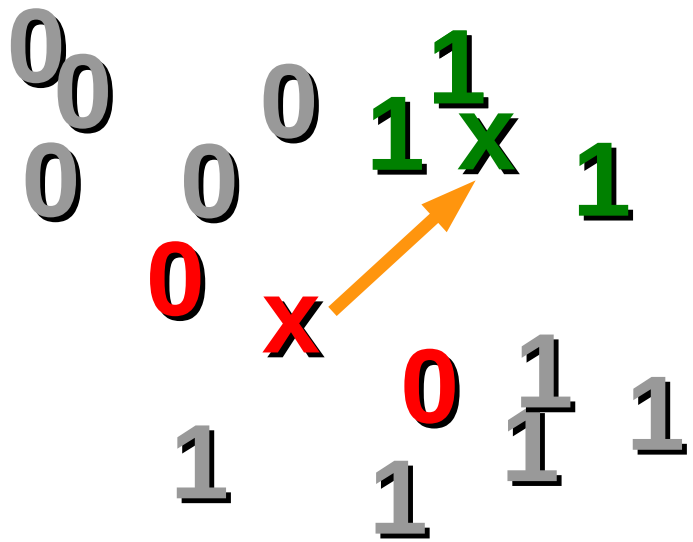
Das Rocchio-Modell: Vollständig

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

- $\alpha = 1, \beta = 0, \gamma = 0 \rightarrow q_{k+1} = q_0$ *(kein Relevance Feedback)*
- Gibt der Nutzer **mehr** Feedback, **erhöhen** wir β und γ
- Häufig werden positive Beispiele stärker gewichtet als negative ($\beta > \gamma$). Typische Werte: $\alpha = 1, \beta = 0.75, \gamma = 0.15$ [1]

Achtung

- Wenn der User nur wenige Dokumente bewertet, drohen Fehler
- Wir bezeichnen diesen Effekt als **Overfitting** *(später mehr)*.



Pseudo Relevance Feedback

- **Problem**: Relevance Feedback erfordert ein händisches Eingreifen seitens des Nutzers.
- Beispiel-**Websuchmaschine**: Nur 4% aller User nutzen Feedback *(oft reicht ein Treffer)* [1].
- Aber: Relevance Feedback **hilft**, den **Recall** zu erhöhen.

Question Answering?

- Recall ist wichtig!
- Aber: Händisches Eingreifen **nicht möglich**.

Ansatz: Pseudo – Relevance Feedback

- Nehme an, der Nutzer hätte die **ersten K** Dokumente der **Trefferliste** als relevant gelabelt.
- Wende dann die oben genannten **Relevance Feedback-Techniken** an.
- Weniger genau als Relevance Feedback *(Lernen von geschätzten Labels!)*, aber **vollautomatisch**!

Query Expansion

- **Relevance Feedback**-Techniken *(siehe oben)* erweitern den Query mit Termen aus den **Ergebnisdokumenten**.
- Gibt es **andere Strategien**, den Query zu erweitern?

Query Expansion

- Wir fügen den Query-Termen **ähnliche Terme** hinzu.

```
'laptop' → 'notebook'
```

Was sind Quellen "ähnlicher" Terme?

- Expansion mit Synonymen aus einem Thesaurus *(z.B. WordNet)*. Problem: Thesauri sind **statisch**.

```
'burning smartphone' → 'samsung'
```

- Statistisches Lernen von Term-Ähnlichkeiten *(später)*.

Outline

Probabilistisches Information Retrieval

- IR bedeutet, **unter Unsicherheit** die "**richtigen**" Dokumente zu finden.
- Für solche Fälle existiert ein mächtiges Werkzeug: Die **Wahrscheinlichkeitsrechnung**.
- Wir betrachten das einfachste **Probabilistische IR-Modell**: Das **Binary Independence Model (BIM)**

Formalisierung

- Dokumente und Queries werden mit Booleschen Vektoren beschrieben: $\mathbf{d} = (d_1, ..., d_n)$ und $\mathbf{q} = (q_1, ..., q_n)$.
- Wir definieren eine **Zufallsvariable R**: Gegeben einen Query **q**, ist Dokument **d** **relevant (R=1)** **oder nicht (R=0)** ?
- **Ansatz**: Ranke Dokumente nach *(absteigender)* **Relevanz-Wahrscheinlichkeit** $P(R = 1 \,|\, \mathbf{d}, \mathbf{q})$.

Probabilistisches IR: Formalisierung

- Wir wenden die **Bayes'sche Regel** an und formen die Relevanz-Wahrscheinlichkeit um:

$$P(R=1|\mathbf{d},\mathbf{q}) = \frac{P(\mathbf{d}|R=1,\mathbf{q}) \cdot P(R=1|\mathbf{q})}{P(\mathbf{d}|\mathbf{q})}$$

- Den **Score** wählen wir so, dass die Dokumente **d** nach **Relevanz-Wahrscheinlichkeit** sortiert werden:

$$\begin{aligned}
s(\mathbf{q},\mathbf{d}) &= \frac{P(R=1|\mathbf{d},\mathbf{q})}{P(R=0|\mathbf{d},\mathbf{q})} && \text{// Bayes'sche Regel} \\
&= \frac{P(\mathbf{d}|R=1,\mathbf{q}) \cdot P(R=1|\mathbf{q}) : \cancel{P(\mathbf{d}|\mathbf{q})}}{P(\mathbf{d}|R=0,\mathbf{q}) \cdot P(R=0|\mathbf{q}) : \cancel{P(\mathbf{d}|\mathbf{q})}} \\
&= \frac{P(\mathbf{d}|R=1,\mathbf{q}) \cdot \cancel{P(R=1|\mathbf{q})}}{P(\mathbf{d}|R=0,\mathbf{q}) \cdot \cancel{P(R=0|\mathbf{q})}} && \text{// beeinflusst nicht das Ranking!} \\
&\propto \frac{P(\mathbf{d}|R=1,\mathbf{q})}{P(\mathbf{d}|R=0,\mathbf{q})}
\end{aligned}$$

Probabilistisches IR: Formalisierung

- $P(\mathbf{d} \mid R = 1, \mathbf{q})$ gibt an, wie typische **relevante Dokumente** aussehen sollten (**d** *gibt an, welche Terme in einem Dokument vorkommen*).
- Wir nehmen an, die einzelnen Terme seien **unabhängig**:

$$P(\mathbf{d}|R = 1, \mathbf{q}) = P(d_1|R = 1, \mathbf{q}) \cdot P(d_2|R = 1, \mathbf{q}) \cdot \ldots \cdot P(d_n|R = 1, \mathbf{q})$$
$$= \prod_{i=1}^{n} P(d_i|R = 1, \mathbf{q})$$

- Die **Scoring-Funktion** ändert sich zu:

$$s(\mathbf{q}, \mathbf{d}) = \frac{P(\mathbf{d}|R = 1, \mathbf{q})}{P(\mathbf{d}|R = 0, \mathbf{q})} = \prod_{i} \frac{P(d_i|R = 1, \mathbf{q})}{P(d_i|R = 0, \mathbf{q})}$$

Probabilistisches IR: Formalisierung

- Wir definieren die **Wahrscheinlichkeiten**, dass ein relevantes (bzw. nicht-relevantes) Dokument einen **Term** t_i enthält:

$$p_i^+ := P(d_i = 1 | R = 1, \mathbf{q})$$
$$p_i^- := P(d_i = 1 | R = 0, \mathbf{q})$$

- **Beispiel**: Query = "Abraham Lincoln", Term t_i = "president"

$$\rightarrow p_i^+ = 0.7, \quad p_i^- = 0.0001$$

- Die **Scoring-Funktion** ändert sich zu:

$$s(\mathbf{q}, \mathbf{d}) = \prod_i \frac{P(d_i | R = 1, \mathbf{q})}{P(d_i | R = 0, \mathbf{q})}$$
$$= \prod_{i:d_i=1} \frac{p_i^+}{p_i^-} \cdot \prod_{i:d_i=0} \frac{1 - p_i^+}{1 - p_i^-}$$

Probabilistisches IR: Formalisierung

Letzte Umformung *(hang in there!)*

- Momentan ist $s(\mathbf{q}, \mathbf{d})$ noch **sehr teuer** zu berechnen *(→ Produkt über alle Terme des Vokabulars)*
- Zusatzannahme: Kommt ein Term t_i **nicht im Query** vor, gilt $p_i^+ = p_i^-$ → der Term **kürzt sich weg**.

$$s(\mathbf{q}, \mathbf{d}) = \prod_{i:d_i=1,q_i=1} \frac{p_i^+}{p_i^-} \cdot \prod_{i:d_i=0,q_i=1} \frac{1-p_i^+}{1-p_i^-}$$

Probabilistisches IR: Schätzung

Schlüsselfrage: Wie bestimmen wir p_i^+ und p_i^-?

Lösung 1: Einfaches Abzählen

$$p_i^+ := \frac{\#\text{ relevante Dokumente die } t_i \text{ enthalten}}{\#\text{ relevante Dokumente}}$$

$$p_i^- := \frac{\#\text{ nicht-relevante Dokumente die } t_i \text{ enthalten}}{\#\text{ nicht-relevante Dokumente}}$$

Lösung 2: Abzählen mit Glättung *(Warum?)*

$$p_i^+ := \frac{\#\text{ relevante Dokumente die } t_i \text{ enthalten} + 0.5}{\#\text{ relevante Dokumente} + 1}$$

$$p_i^- := \frac{\#\text{ nicht-relevante Dokumente die } t_i \text{ enthalten} + 0.5}{\#\text{ nicht-relevante Dokumente} + 1}$$

Probabilistisches IR: Schätzung

Problem

Wir wissen **nicht welche** Dokumente relevant (bzw. nicht-relevant) sind! Wie **berechnen** wir p_i^+ und p_i^-?

Lösung für nicht-relevante Dokumente (p_i^-)?

- **Idee**: **Fast alle** Dokumente sind nicht-relevant!
- Approximation *(sehr leicht zu berechnen!)*

$$p_i^- := \frac{\#\ \textbf{alle}\ \text{Dokumente die}\ t_i\ \text{enthalten}}{\#\ \textbf{alle}\ \text{Dokumente}}$$

Lösung für relevante Dokumente (p_i^+)?

- **Idee**: Terme die **im Query vorkommen** sind etwas häufiger!
- Approximation *(auch leicht zu berechnen!)*

$$p_i^+ := \begin{cases} \frac{1}{3} + \frac{2}{3} \cdot \frac{\#\ \text{Dokumente mit}\ t_i}{\#\ \textbf{alle}\ \text{Dokumente}} & \text{falls}\ t_i\ \text{im Query vorkommt} \\ p_i^- & \textit{sonst} \end{cases}$$

Probabilistisches IR: Relevance Feedback

Lösung 2: Relevance Feedback

- Es gibt eine **noch bessere Lösung**, um p_i^+ zu schätzen: **Relevance Feedback**
- Nach seiner Suche **markiert** der Benutzer eine **Menge von Dokumenten D_R** als **relevant**.
- **Idee**: Kommt ein Term **häufig** in den Dokumenten D_R vor, sollte seine Wahrscheinlichkeit p_i^+ **steigen**.
- Wir **verfeinern** unsere Schätzung für p_i^+:

$$p_i^+ \leftarrow (1-\alpha) \cdot p_i^+ + \alpha \cdot \frac{\#\text{ Dokumente in } D_R \text{ die } t_i \text{ enthalten}}{\#\ D_R}$$

Outline

PageRank

Idee

- Verwende die Link-Struktur zwischen Dokumenten, um die Genauigkeit von Information Retrieval zu verbessern.
- **Ein Link ist ein Indikator für die Autorität des Link-Ziels!**

Beispiele

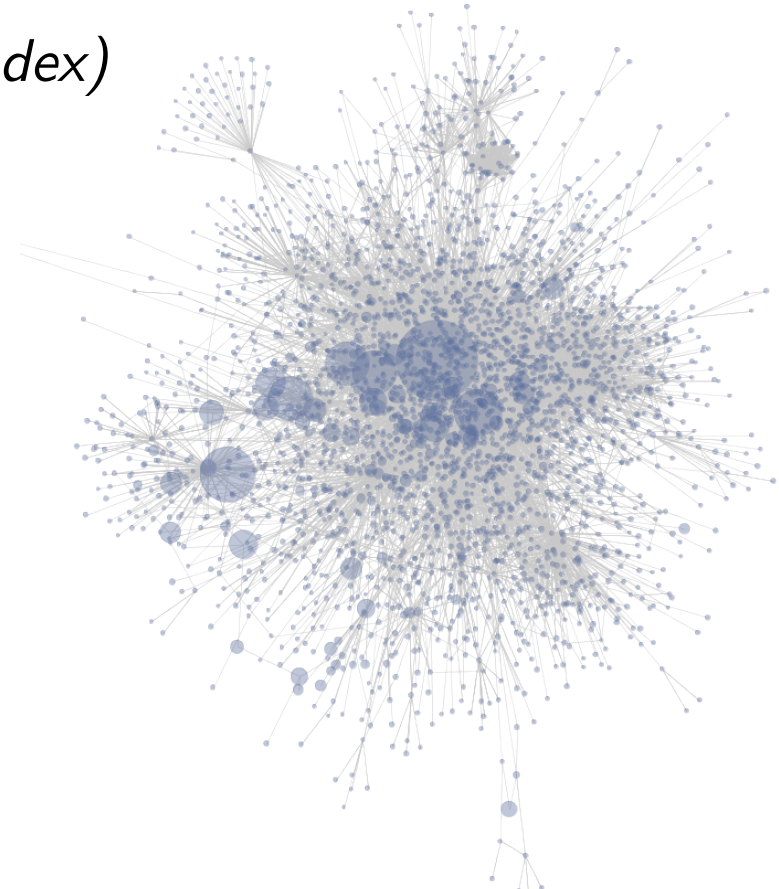
- Wissenschaftliches Zitieren *(H-Index)*
- Anzahl der Follower auf Twitter

Websites/Dokumente, die häufig gelinkt werden...

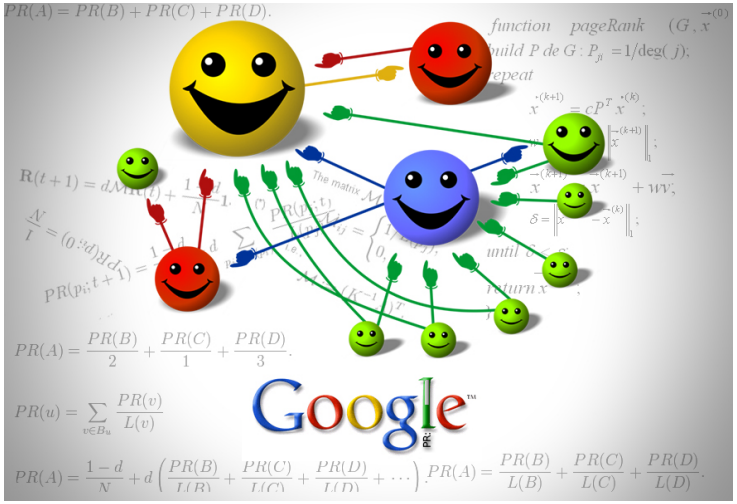
... besitzen eine "hohe Autorität"

... erhalten höhere Scores

... werden durch Crawler häufiger besucht.



PageRank: Einführung



- Im Folgenden: Der **PageRank** [3] als ein populäres Maß für Autorität
- Dieser dient als Parameter in Suchmaschinen *(→ Boost-Faktor für Dokument-Scores)*

Ansatz 1

- Autorität = **Anzahl** der eingehenden Links
- Problem: Links von **“wichtigen” Seiten** sollten **bedeutender** sein als Links von “unwichtigen” Seiten
- Problem: Anfälligkeit für **Spam** *(Erzeugung künstlicher Links, um den eigenen PageRank zu optimieren)*

Ansatz 2

- “Wird eine Seite von vielen wichtigen Seiten verlinkt?”
- **Problem** (“Henne-Ei”): Um Seite X zu bewerten, müssen wir die Bewertung von Seiten Y kennen.

Das Random Surfer Modell

Ansatz 2: Formalisierung

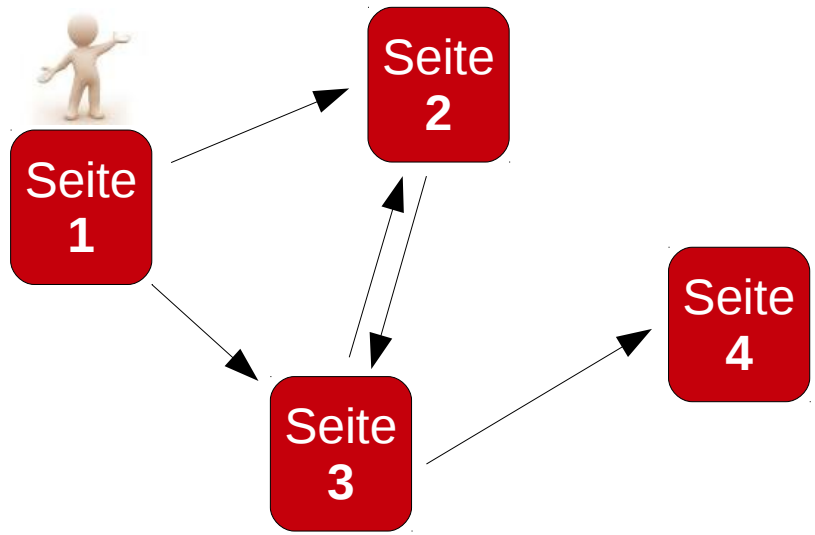
- Eine Seite ist wichtig, wenn sich ein **zufällig surfender Webnutzer** (**"random surfer"**) oft dort aufhält.
- Der Nutzer beginnt bei einer **zufälligen Seite** p_0
- Der Nutzer **bewegt sich** iterativ von einer Seite p_n zur nächsten (p_{n+1}), indem er einem zufälligen von p_n ausgehenden **Link folgt**.
- Alle ausgehenden Links einer Seite sind dabei gleich wahrscheinlich.

Schlüsselfrage

- Wie ist der Aufenthaltsort des zufälligen Surfers über die Webseiten **verteilt**?

Das Random Surfer Modell: Beispiel

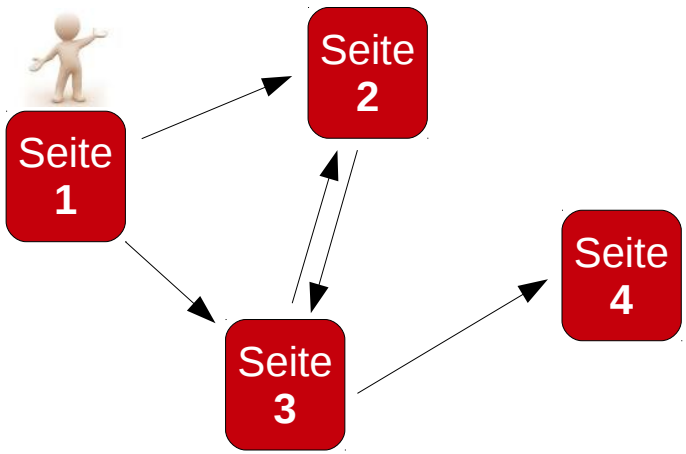
- Der Nutzer beginnt bei Seite 1
- Nach einem Schritt:
  - ... ist er mit 50% Wahrscheinlichkeit bei Seite 2
  - ... ist er mit 50% Wahrscheinlichkeit bei Seite 3
- Nach zwei Schritten...?
- Nach drei Schritten...?



Das Random Surfer Modell: Formalisierung

- Ein **Link** von Seite j zu Seite i ist ein **Tupel** (j, i)
- Wir bezeichnen die **Menge aller Links** mit $\mathbb{L}$.
- $\mathbb{L}$ definiert eine Übergangsmatrix T:

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

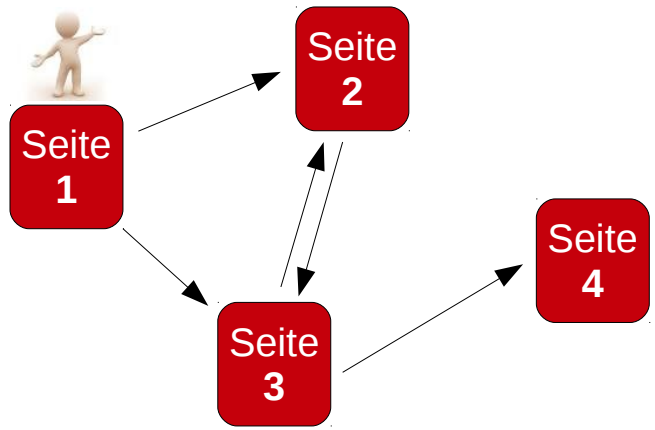


Ein User auf Seite 3 bewegt sich mit Wahrscheinlichkeit 1/2 zu Seite 2

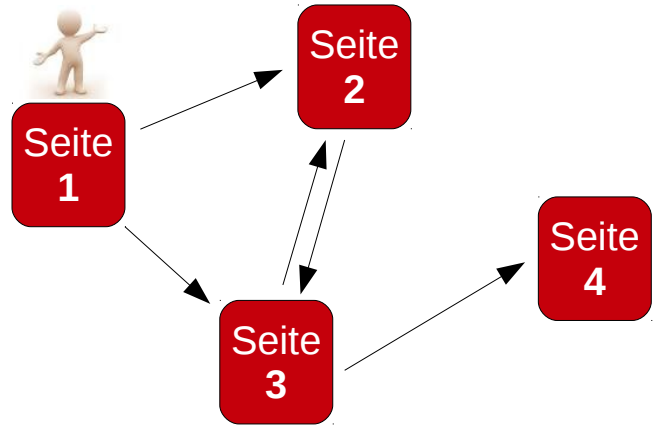
$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \end{pmatrix}$$

Das Random Surfer Modell: Formalisierung

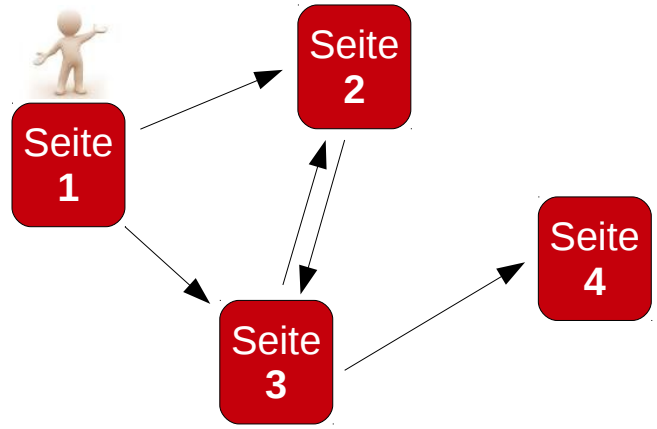
- Nach **n Schritten** befindet sich der Surfer mit einer bestimmten **Wahrscheinlichkeit** auf jeder Seite.
- Wir fassen die Wahrscheinlichkeiten in einem **Vektor** $\mathbf{p}_n$ zusammen *(wo ist der Nutzer gerade mit welcher Wahrscheinlichkeit?)*



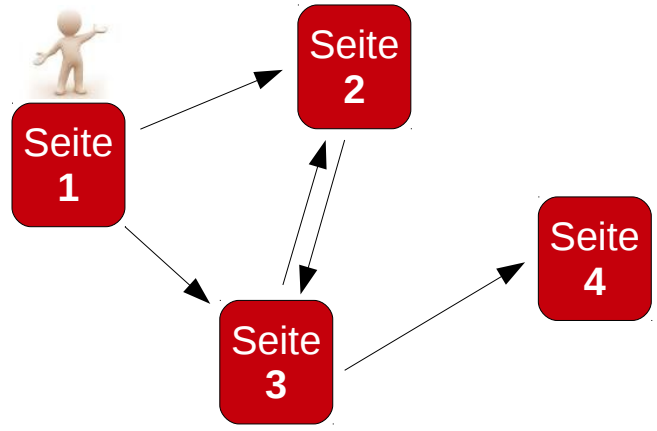
Das Random Surfer Modell: Formalisierung



Das Random Surfer Modell: Formalisierung



Das Random Surfer Modell: Formalisierung



Das Random Surfer Modell: Formalisierung

- **Problem**: **Sackgassen** (Page 4) → Folgezustand undefiniert
- **Trick**: **"Teleporting"** → Wir erlauben zufällige gleichverteilte Sprünge zu einer beliebigen anderen Seite.

1. **Sackgasse**: Springe zufällig zu **beliebiger** Seite.
2. **nicht in Sackgasse**: Springe mit Wahrscheinlichkeit α zu beliebiger anderer Seite.

Geglättete Übergangsmatrix T' (sei k die Anzahl der Seiten)

$$T'_{ij} := \begin{cases} 1/k & \text{falls Seite } j \text{ Sackgasse} \\ (1-\alpha) \cdot T_{ij} + \alpha \cdot 1/k & \textit{sonst} \end{cases}$$

Beispiel ($\alpha = 0.1$)

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

Markov-Ketten

Wir nennen unser mathematisches Modell eine **Markov-Kette**:

Definition (Markov-Kette)

Gegeben sei ein Zustandsraum $S = \{1, 2, ..., n\}$. *Ein stochastischer Prozess generiert eine Sequenz von Zuständen* $S_1, S_2, S_3, ... \in S$, *so dass für alle* $n \in \mathbb{N}$ *gilt:*

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

Dann nennen wir den Prozess eine **diskrete, endliche Markov-Kette erster Ordnung**.

Anmerkungen

- Die Wahrscheinlichkeiten $P(S_{n+1} = j | S_n = i)$ entsprechen den Einträgen unserer Übergangsmatrix T'_{ij} *(siehe oben)*.

Markov-Ketten

Unsere spezielle Markov-Kette ist ...

...irreduzibel:

- Wir können jeden Zustand (d.h. jede Seite) von jedem anderen Zustand (d.h. von jeder anderen Seite) aus erreichen.

... aperiodisch:

- Es gibt für jeden Zustand mindestens zwei Zyklen teilerfremder Länge.

Beide Eigenschaften...

- ... sind durch das Teleporting garantiert (warum?)

Markov-Ketten: Gekritzel

Random Surfer: Konvergenz

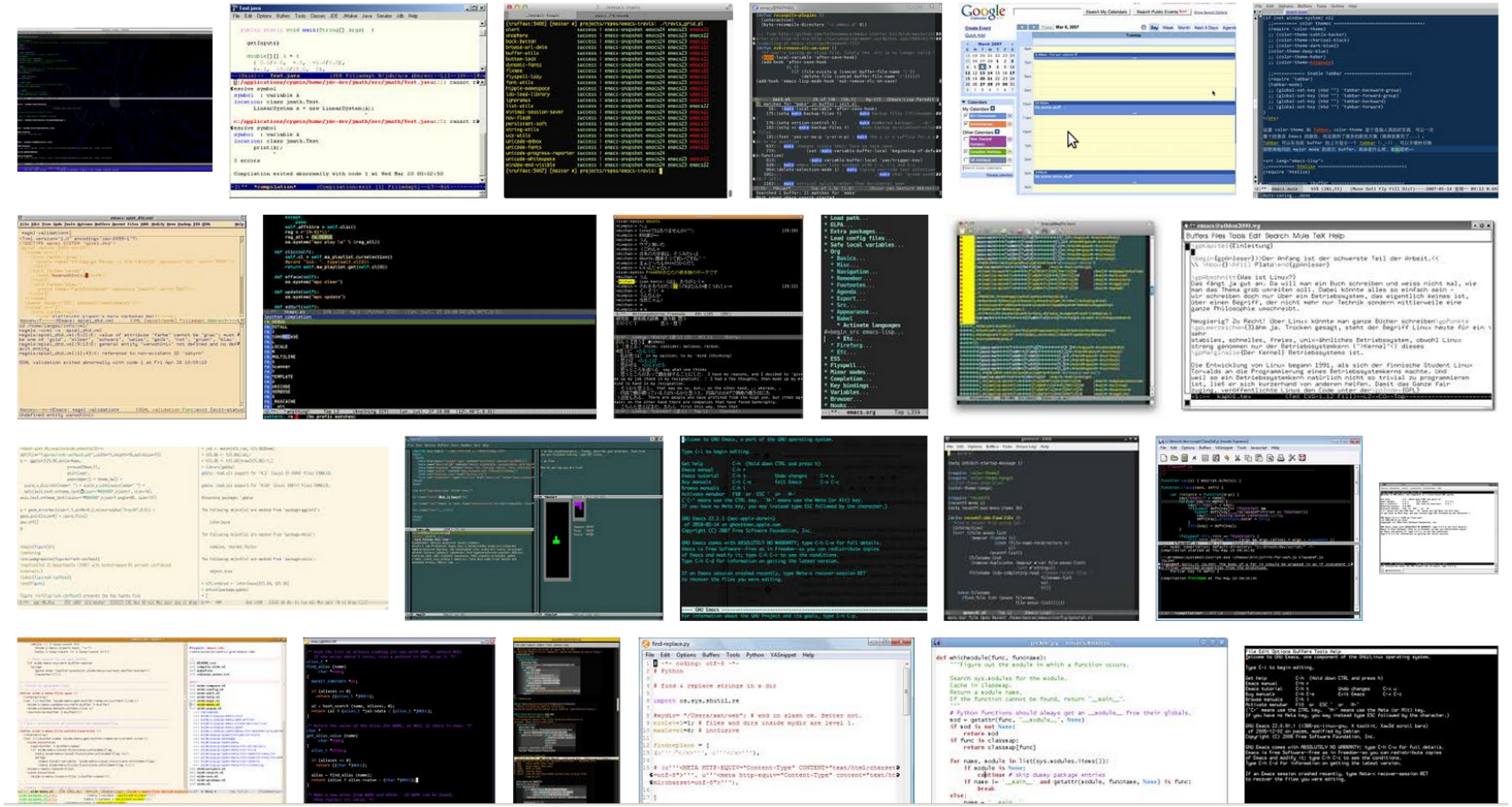
Für **irreduzible, aperiodische Markov-Ketten** gilt:

- Die Folge $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots$ konvergiert gegen den **Grenzwert** $\mathbf{p}$.
- **Umgangssprachlich ausgedrückt**: "Wenn der Random Surfer **unendlich lange** surft, erhalten wir eine fixe **Verteilung seines Aufenthaltsortes** über die Seiten".
- Dieser Grenzwert $\mathbf{p}$ sollte sich nicht ändern, wenn der Surfer **einen Schritt** macht, d.h.

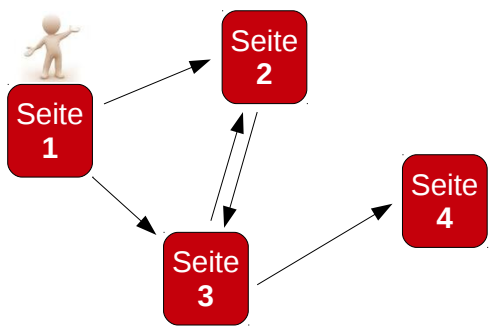
$$T \cdot \mathbf{p} = \mathbf{p}$$

- Wir sehen: $\mathbf{p}$ ist ein **Eigenvektor** von T!
- Wir nennen $\mathbf{p}$ auch die **stationäre Verteilung** von T'
- $\mathbf{p}$ hängt nicht von $\mathbf{p}_1$ ab *(d.h., es ist egal, wo der random surfer startet)*.

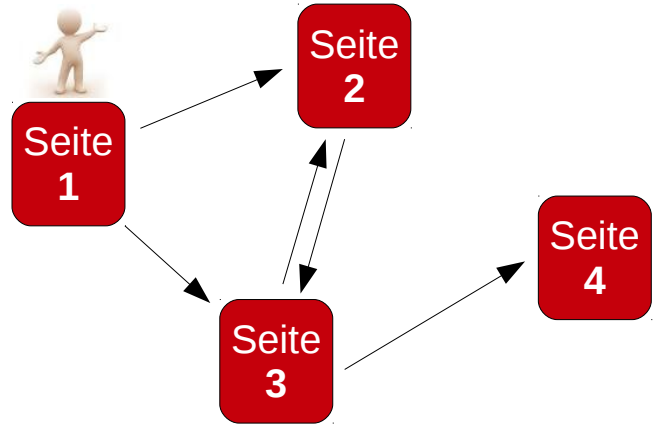
Random Surfer: Beispiel



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



Random Surfer: Konvergenz



Der "PageRank" einer Seite...

- ... ist nichts anderes als der zugehörige Eintrag in der stationären Verteilung $\mathbf{p}$.
- Wir berechnen $\mathbf{p}$, indem wir *(ausgehend von einem zufälligen Startvektor* $\mathbf{p}_1$*)* immer wieder mit T multiplizieren *(d.h., wir lassen den Surfer immer wieder einen Schritt durchführen)*.

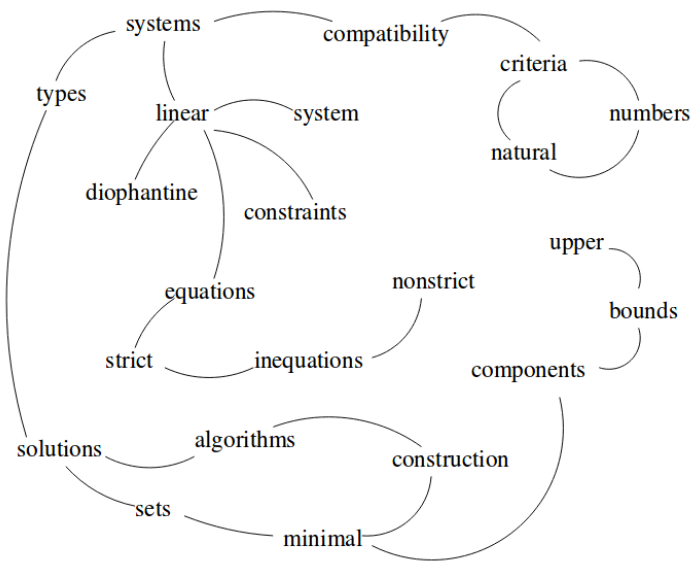
$$\mathbf{p} = \lim_{k \to \infty} T^k \cdot \mathbf{p}_1 = \ldots \cdot T \cdot T \cdot T \cdot \mathbf{p}_1 = \begin{pmatrix} 0.08164946 \\ 0.28851762 \\ 0.37805757 \\ 0.25177536 \end{pmatrix}$$

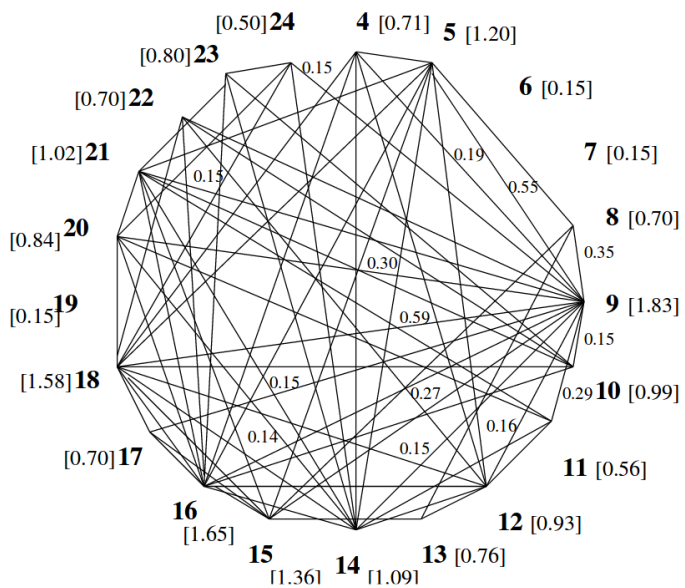
Interpretation?

- Seite 1 hat keine Links → hier ist der User selten.
- Seite 3 hat die wertvollsten eingehenden Links.
- Seite 2 hat (im Vergleich mit Seite 4) noch einen zusätzlichen Link, und somit einen etwas höheren PageRank.

Ausblick: TextRank

- Wir können mit PageRank **generelle Link-Strukturen** analysieren!
- **Beispiel**: **Wörter** und **Sätze** innerhalb eines Texts
- **Wörter** *(links unten)* werden verlinkt
  falls sie im *gleichen Kontext* auftauchen
- **Sätze** *(rechts unten)* werden verlinkt
  falls sie *ähnliche Worte* enthalten.
- PageRank (hier: *TextRank* [2]) generiert **Schlüsselterme** und **Zusammenfassungen** von Texten





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.