Advanced Topics in Information Retrieval

Klaus Berberich
(kberberi@mpi-inf.mpg.de)

Winter Semester 2014/2015
Saarland University

Outline

0.1. Organization

0.2. Documents & Queries

0.3. Retrieval Models

0.4. Link Analysis

0.5. Indexing & Query Processing

0.6. Effectiveness Measures

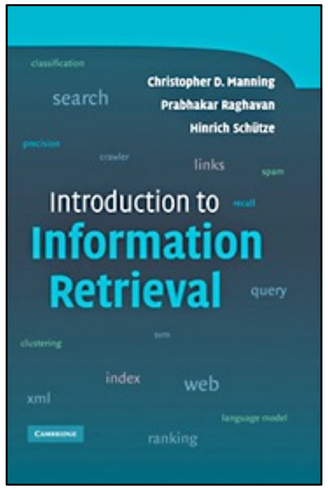
0.1. Organization

- **Lectures** on **Monday 10:15–11:45** in **R024/E1.4** (MPI-INF)
- **Tutorials** on **Monday 14:15–15:45** in **R023/E1.4** (MPI-INF)

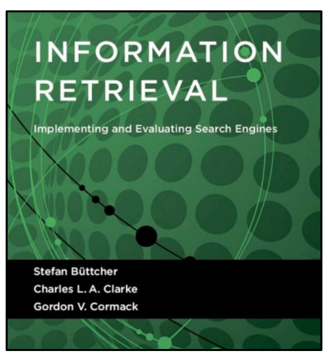
- Lecturer: **Klaus Berberich** (kberberi@mpi-inf.mpg.de)
  - Office hours on **Monday 13:00–14:00** (or appointment by e-mail)
- Tutor: **Dhruv Gupta** (dhgupta@mpi-inf.mpg.de)

- Prerequisite: Successful participation in the core course **Information Retrieval & Data Mining** or equivalent one

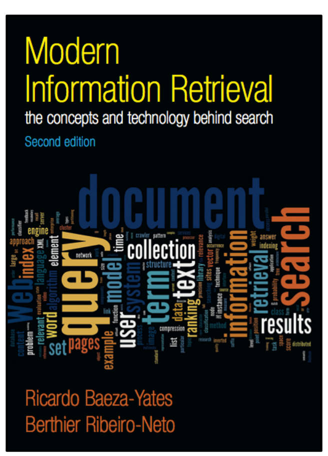
Background Literature



- **C. D. Manning, P. Raghavan, H. Schütze,**
  Introduction to Information Retrieval,
  Cambridge University Press, 2008
  http://www.informationretrieval.org



- **S. Büttcher, C. L. A. Clarke, G. V. Cormack,**
  Information Retrieval,
  MIT Press, 2010



- **R. Baeza-Yates and R. Ribeiro-Neto,**
  Modern Information Retrieval,
  Addison-Wesley, 2011

Agenda (2014)

1. Social Media
2. Recommender Systems
3. Semantics
4. Personalization
5. Efficiency & Scalability
6. Novelty & Diversity

Agenda (2015)

7. Learning to Rank
8. Dynamics & Age
9. Mining & Organization
10. Evaluation

Exercise Sheets & Tutorials

- **Biweekly exercise sheets**
  - **six** exercise sheets each with **up to six problems**
  - handed out during the lecture on **Monday**
  - due by **Thursday 11:59 PM** of the following week
  - submit **electronically as PDF** to atir2014@mpi-inf.mpg.de (best: typeset using LaTeX, worst: scans of your handwriting)
- **Biweekly tutorials**
  - on Mondays after due dates
  - we'll grade your solutions as (**P**)resentable, (**S**)erious, (**F**)ail
  - **no example solutions**

Obtaining 6 ECTS

- Submit serious or better solutions to **at least 50%** of problems
- **Present** solutions in tutorial
  - **at least once** during the semester
  - additional presentations score you **bonus points**
    (one grade per bonus point, at most three, at most one per session)
- Pass **oral exam** at the end of the semester

Registration & Password

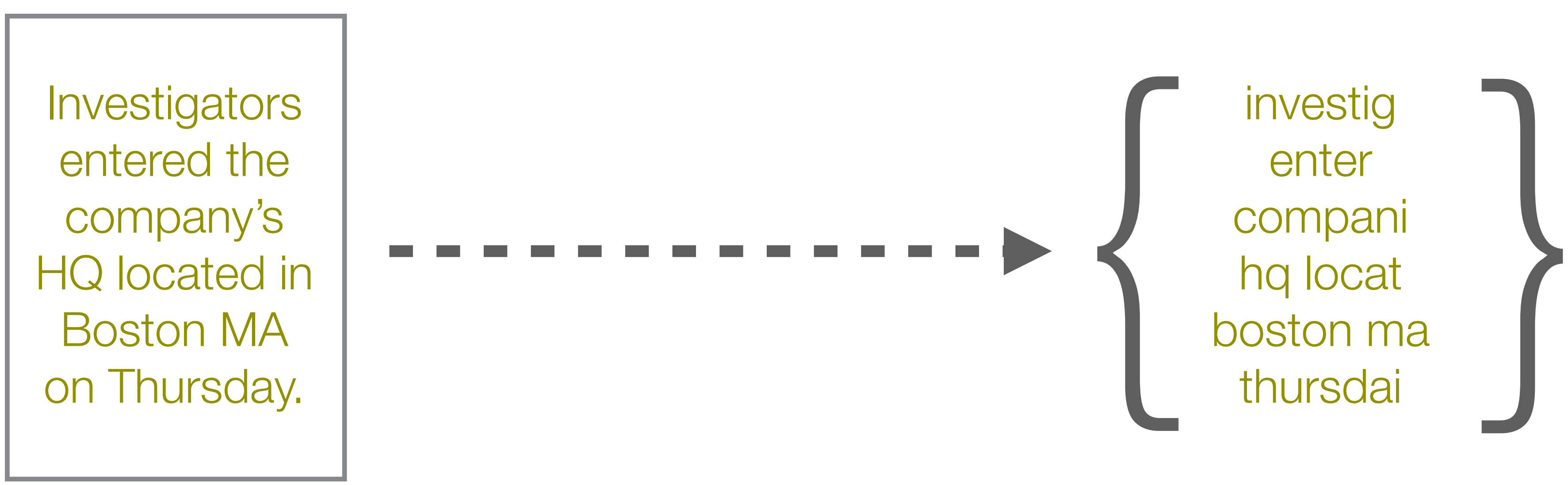
- You'll have to register for this course and the exam in **HISPOS**
- Please let us also know that you attend this course and send an e-mail with subject **"Registration"** to atir2014@mpi-inf.mpg.de
  - Full name
  - Student number
  - Preferred e-mail address
- Some materials (e.g., papers and data) will be made available in a password-protected area on the course website
  - Username: atir2014 / Password: < first eight digits of π >

Questions? Ideas? Requests?

0.2. Documents & Queries

- Pre-processing of documents and queries typically includes
  - **tokenization** (e.g., splitting them up at white spaces and hyphens)
  - **stemming** or **lemmatization** (to group variants of the same word)
  - **stopword removal** (to get rid of words that bear little information)

- This results in a **bag (or sequence) of indexable terms**



0.3. Retrieval Models

- Retrieval model defines for a given **document collection** D and a **query** q which documents to return in which order
  - **Boolean retrieval**
  - Probabilistic retrieval models (e.g., binary independence model)
  - **Vector space model** with **tf.idf** term weighting
  - **Language models**
  - Latent topic models (e.g., LSI, pLSI, LDA)

Boolean Retrieval

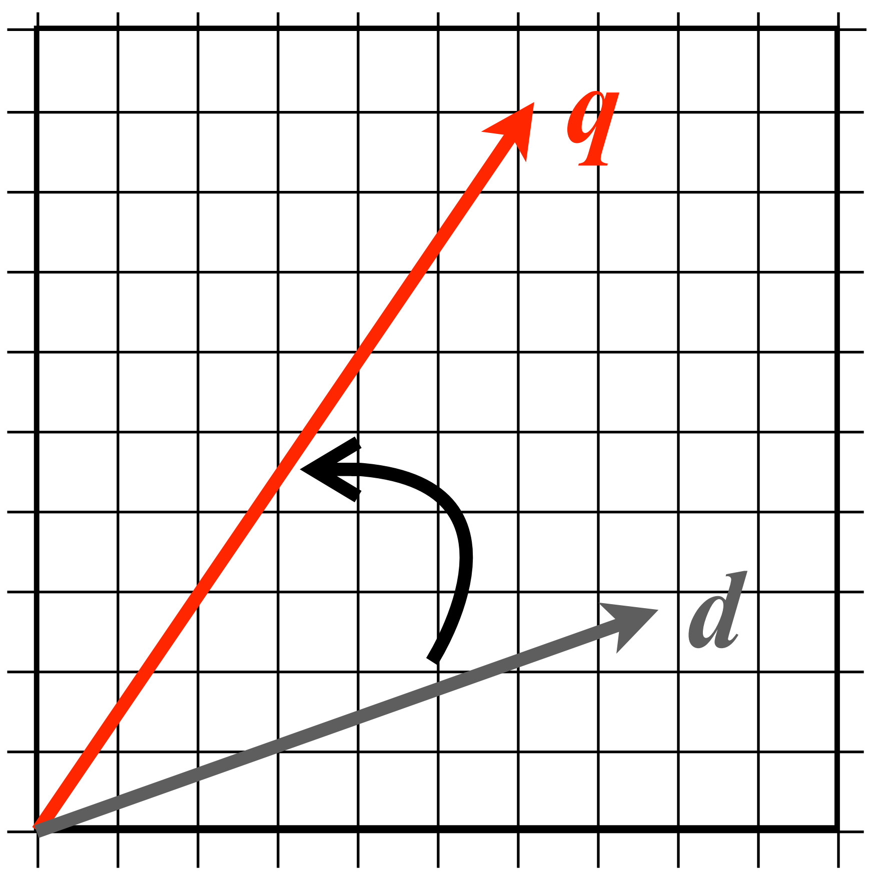
- **Boolean variables** indicate presence/absence of query terms
- **Boolean operators** `AND`, `OR`, and `NOT`
- Boolean queries are **arbitrary compositions** of those, e.g.:
  - brutus `AND` caesar `AND NOT` calpurnia
  - `NOT` ((duncan `AND` macbeth) `OR` (capulet `AND` montague))
  - …
- **Query result** is the **(unordered) set** of documents satisfying (i.e., "matching") the query
- **Extensions of Boolean retrieval** (e.g., proximity, wildcards, fields) with **rudimentary ranking** (e.g., weighted matches) exist

Vector Space Model

- Vector space model considers **queries and documents** as vectors in a common **high-dimensional vector space**

- Cosine similarity between two vectors **q** and **d** is the **cosine of the angle between them**

$$
\begin{aligned}
sim(q,d) &= \frac{q \cdot d}{\|q\| \, \|d\|} \\
&= \frac{\sum_v q_v \, d_v}{\sqrt{\sum_v q_v^2} \sqrt{\sum_v d_v^2}} \\
&= \frac{q}{\|q\|} \cdot \frac{d}{\|d\|}
\end{aligned}
$$



tf.idf

- How to set the **components** of query and document vectors?
- Intuitions behind **tf.idf term weighting**:
  - documents should profit if they **contain a query term more often**
  - **query terms** should be **weighted** (e.g., snowden documentation)
- **Term frequency** tf(v,d) – # occurrences of term v in document **d**
- **Document frequency** df(v) – # documents containing term v
- Components of **document vectors** set as

$$d_v = tf(v,d) \log \frac{|D|}{df(v)}$$

- Components of **query vectors** set as binary indicators

Language Models

- Language model describes the **probabilistic generation** of elements from a **formal language** (e.g., sequences of words)
- Documents and queries can be seen as **samples from a language model** and be used to **estimate its parameters**

$$P[\, v \mid \theta_d \,] = \frac{tf(v,d)}{\sum_w tf(w,d)}$$

a b a c a
a a c a b
b b b a a
c b a a a
a a a a a

$$P[\, a \mid \theta_d \,] = \frac{16}{25}$$

$$P[\, b \mid \theta_d \,] = \frac{6}{25}$$

$$P[\, c \mid \theta_d \,] = \frac{3}{25}$$

Smoothing

- Terms that do **not occur** in a document have **zero probability** of being generated by the estimated language model

- Parameter estimation from a **single document or query** bears the **risk of overfitting** to this very limited sample

- Smoothing methods estimate parameters considering the **entire document collection as a background model**

Smoothing

- **Jelinek-Mercer smoothing**

$$P[\, v \mid \theta_d \,] = \alpha \cdot \frac{tf(v,d)}{\sum_w tf(w,d)} + (1-\alpha) \cdot \frac{tf(v,D)}{\sum_w tf(w,D)}$$

- **Dirichlet smoothing**

$$P[\, v \mid \theta_d \,] = \frac{tf(v,d) + \mu \, \frac{tf(v,D)}{\sum_w tf(w,D)}}{\sum_w tf(w,d) + \mu}$$

- Smoothing **eliminates zero probabilities** and introduces a **relative term weighting** (idf-like effect) since more common terms now have higher probability for all documents

Query Likelihood vs. Divergence

- **Query-likelihood approaches** rank documents according to the probability that their language model generates the query

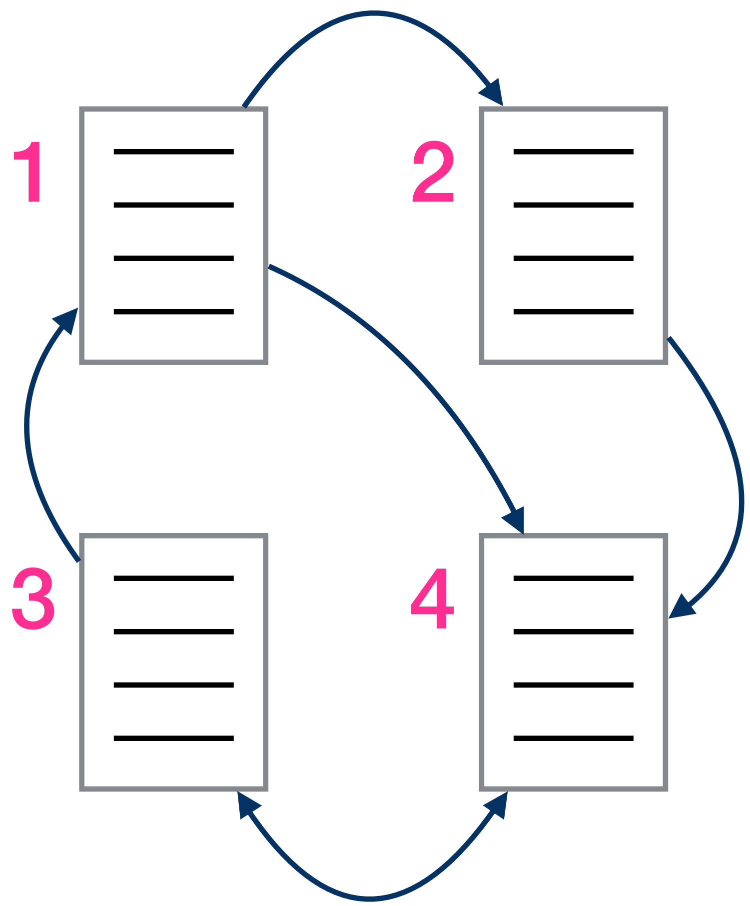
$$P[\,q \mid \theta_d\,] \propto \prod_{v \in q} P[\,v \mid \theta_d\,]$$

- **Divergence-based approaches** rank according to the **Kullback-Leibler divergence** between the query language model and language models estimate from documents

$$KL(\,\theta_q \,\|\, \theta_d) = \sum_{v} P[\,v \mid \theta_q\,] \, \log \, \frac{P[\,v \mid \theta_q\,]}{P[\,v \mid \theta_d\,]}$$

0.4. Link Analysis

- Link analysis methods consider the Web's **hyperlink graph** to determine **characteristics** of individual **web pages**



$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- They can also be applied to graph structures obtained from **other kinds of data** (e.g., social networks and word co-occurrence

PageRank

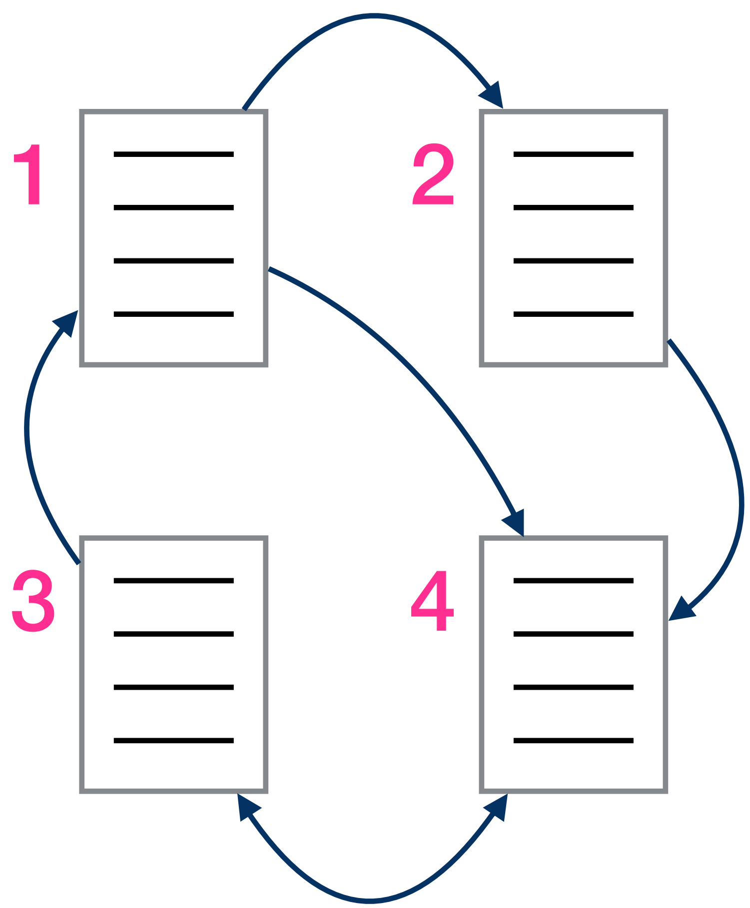
- PageRank (by Google) is based on the following **random walk**
  - jump to a random vertex (1 / |V|) in the graph with probability ε
  - follow a random outgoing edge (**1 / out(v)**) with probability (1-ε)

$$p(v) = (1 - \epsilon) \cdot \sum_{(u,v) \in E} \frac{p(u)}{out(u)} + \frac{\epsilon}{|V|}$$

- PageRank score **p(v)** of vertex v is a **measure of popularity** and corresponds to its **stationary visiting probability**

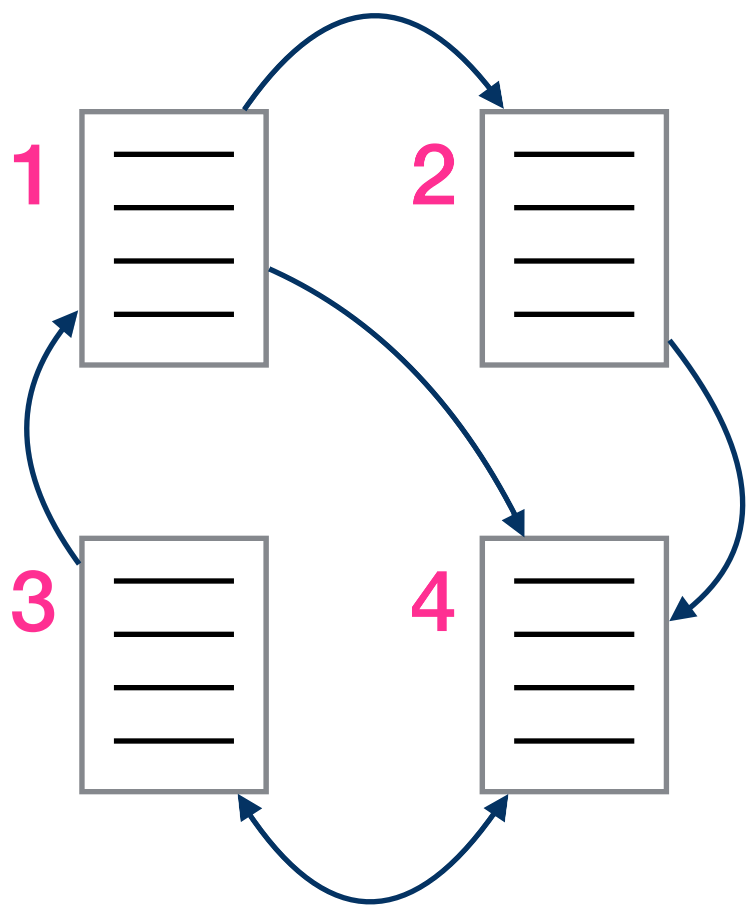
PageRank

- PageRank scores correspond to components of the **dominant Eigenvector π** of the **transition probability matrix** P which can be computed using the **power-iteration method**



PageRank

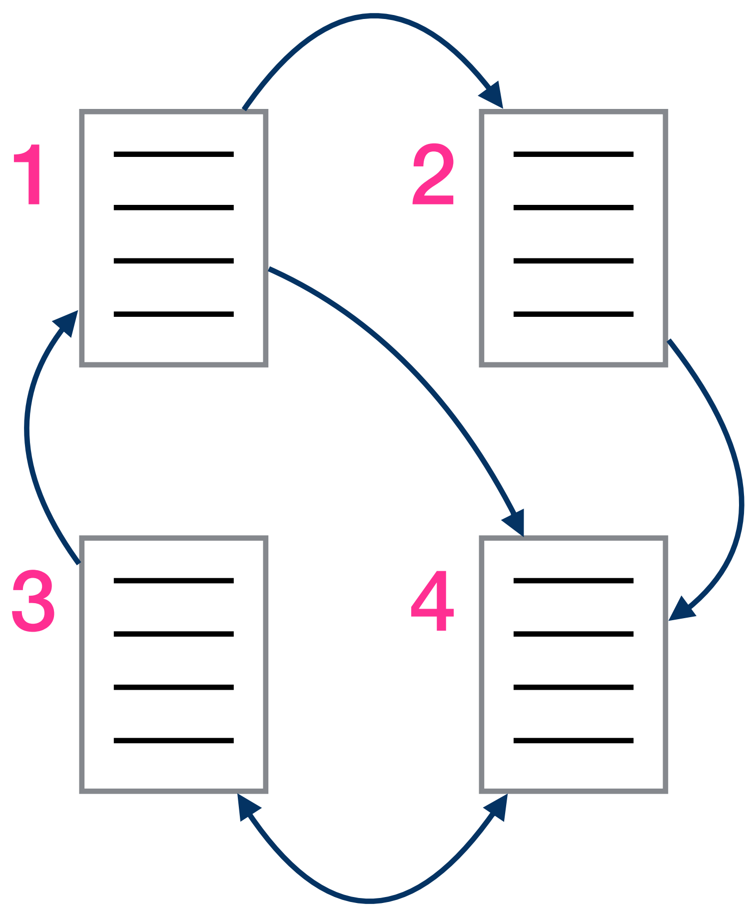
- PageRank scores correspond to components of the **dominant Eigenvector π** of the **transition probability matrix** P which can be computed using the **power-iteration method**



$$\epsilon = 0.8$$

PageRank

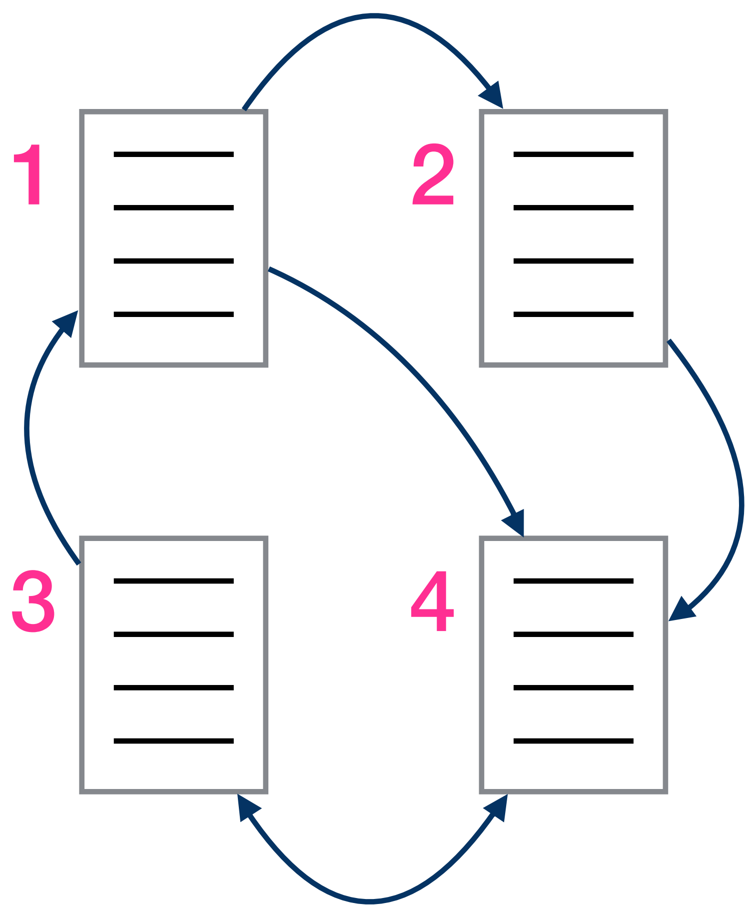
- PageRank scores correspond to components of the **dominant Eigenvector π** of the **transition probability matrix** P which can be computed using the **power-iteration method**



$$\epsilon = 0.8 \qquad P = \begin{bmatrix} 0.05 & 0.45 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.05 & 0.85 \\ 0.45 & 0.05 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.85 & 0.05 \end{bmatrix}$$

PageRank

- PageRank scores correspond to components of the **dominant Eigenvector π** of the **transition probability matrix** P which can be computed using the **power-iteration method**

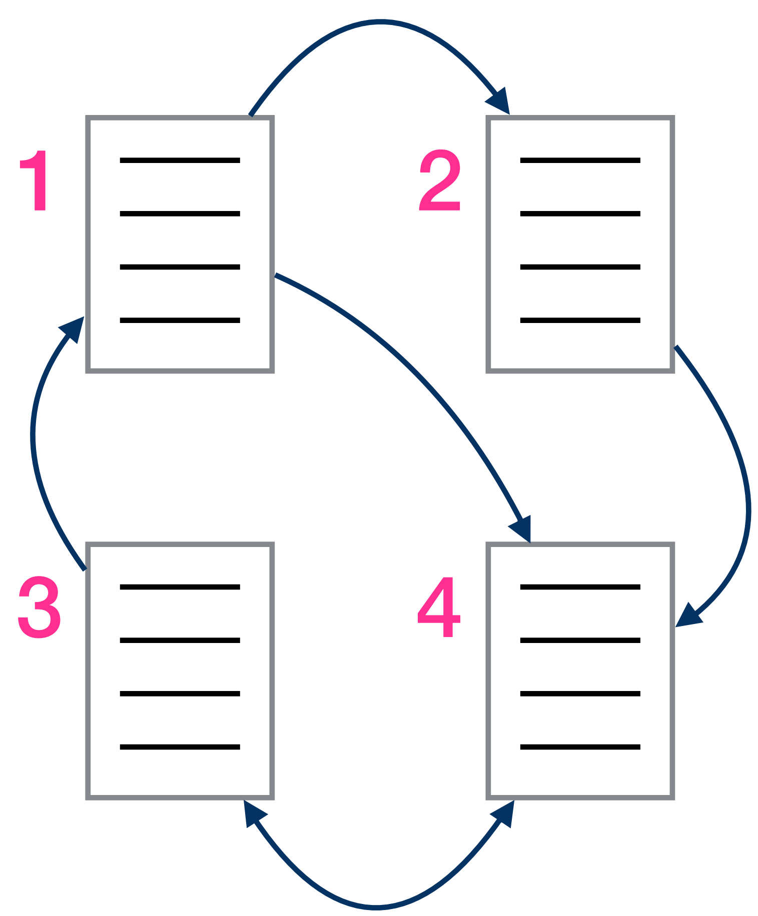


$$\epsilon = 0.8 \qquad P = \begin{bmatrix} 0.05 & 0.45 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.05 & 0.85 \\ 0.45 & 0.05 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.85 & 0.05 \end{bmatrix}$$

$$\pi^{(0)} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

PageRank

- PageRank scores correspond to components of the **dominant Eigenvector π** of the **transition probability matrix** P which can be computed using the **power-iteration method**



$$\epsilon = 0.8 \qquad P = \begin{bmatrix} 0.05 & 0.45 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.05 & 0.85 \\ 0.45 & 0.05 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.85 & 0.05 \end{bmatrix}$$

$$\begin{aligned} \pi^{(0)} &= \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix} \\ \pi^{(1)} &= \begin{bmatrix} 0.15 & 0.15 & 0.25 & 0.45 \end{bmatrix} \\ \pi^{(2)} &= \begin{bmatrix} 0.15 & 0.11 & 0.41 & 0.33 \end{bmatrix} \\ &\vdots \\ \pi^{(10)} &= \begin{bmatrix} 0.18 & 0.12 & 0.34 & 0.36 \end{bmatrix} \end{aligned}$$

HITS

- Hyperlink-Inducted Topics Search (HITS) operates on a **subgraph of the Web** induced by a keyword query and considers
  - **hubs** as vertices **pointing to good authorities**
  - **authorities** as vertices **pointed to by good hubs**
- **Hub score** h(u) and **authority score a(v)** defined as

$$h(u) \propto \sum_{(u,v) \in E} a(v) \qquad a(v) \propto \sum_{(u,v) \in E} h(u)$$

- Hub vector h and authority vector a are **Eigenvectors** of the **co-citation matrix** AA^T and **co-reference matrix** A^TA

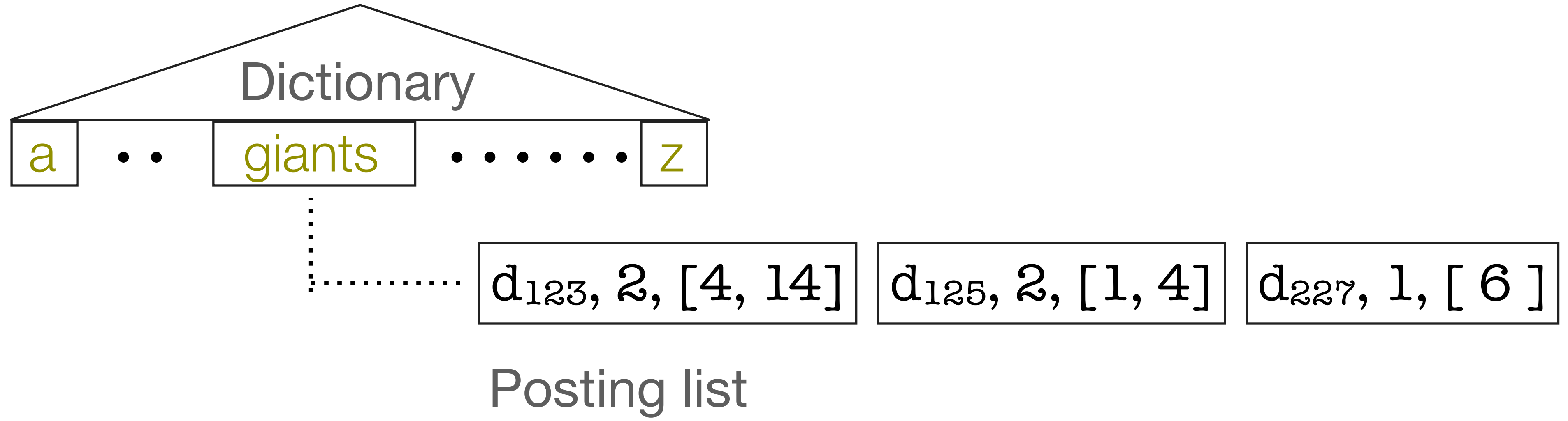
$$h = \alpha\, \beta A A^T h \qquad a = \alpha\, \beta A^T A\, a$$

0.5. Indexing & Query Processing

- Retrieval models define which documents to return for a query but **not how they can be identified efficiently**
- **Index structures** are an essential building block for IR systems; variants of the inverted index are by far most common
- **Query processing methods** operate on these index structures
  - **holistic** query processing methods determine **all query results** (e.g., term-at-a-time, document-at-a-time)
  - **top-k** query processing methods determine the **best k query results** (e.g., WAND, BMW, Fagin's TA & NRA)

Inverted Index

- Inverted index as widely used index structure in IR consists of
  - **dictionary** mapping terms to term identifiers and statistics (e.g., df)
  - **posting list** for every term recording details about its occurrences
- Posting lists can be **document- or score-ordered** and be equipped with additional structure (e.g., to support **skipping**)
- Postings contain a **document identifier** plus additional **payloads** (e.g., term frequency, tf.idf score contribution, term offsets)



Posting-List Compression

- It is often **faster to read and decompress data**, both from main memory and secondary storage, **than to read it uncompressed**
- **Posting lists** of an inverted index are **typically compressed**
  - **delta encoding** for sequences of non-decreasing integers (e.g., document identifiers or term offsets)

    $\langle 1, 7, 11, 21, 42, 66 \rangle \dashrightarrow \langle 1, 6, 4, 10, 21, 24 \rangle$

  - **variable-byte encoding** (aka. 7-bit encoding) represents integers (e.g., deltas of term offsets) as sequences of **1 continuation + 7 data bits**

    314 = 00000000 00000000 00000001 00111010

    $\dashrightarrow$ 00000010 10111010

Term-at-a-Time

- Processes posting lists for query terms $\langle$ q$_1$,…,q$_m$ $\rangle$ **one at a time**
- Maintains an **accumulator for each document** seen; after processing the first **k** query terms this corresponds to

$$acc(d) = \sum_{i=1}^{k} score(q_i, d)$$

a ⋯⋯ d_1, 0.2 | d_3, 0.1 | d_5, 0.5

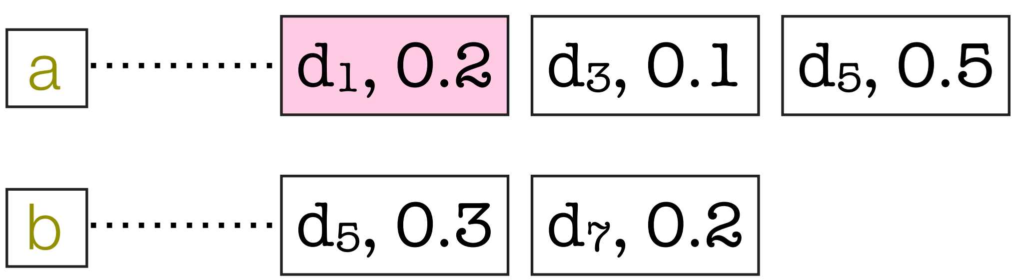
b ⋯⋯ d_5, 0.3 | d_7, 0.2

- **Main memory** proportional to number of accumulators
- Top-k result determined at the end by sorting accumulators

Term-at-a-Time

- Processes posting lists for query terms ⟨ $q_1,\ldots,q_m$ ⟩ **one at a time**
- Maintains an **accumulator for each document** seen; after processing the first **k** query terms this corresponds to

$$acc(d) = \sum_{i=1}^{k} score(q_i, d)$$



- **Main memory** proportional to number of accumulators
- Top-k result determined at the end by sorting accumulators

Term-at-a-Time

- Processes posting lists for query terms ⟨ $q_1,\ldots,q_m$ ⟩ **one at a time**

- Maintains an **accumulator for each document** seen; after processing the first **k** query terms this corresponds to

$$acc(d) = \sum_{i=1}^{k} score(q_i, d)$$

a ⋯⋯ | d_1, 0.2 | d_3, 0.1 | d_5, 0.5 |

b ⋯⋯ | d_5, 0.3 | d_7, 0.2 |

- **Main memory** proportional to number of accumulators

- Top-k result determined at the end by sorting accumulators

Term-at-a-Time

- Processes posting lists for query terms $\langle$ q$_1$,…,q$_m$ $\rangle$ **one at a time**

- Maintains an **accumulator for each document** seen; after processing the first **k** query terms this corresponds to

$$acc(d) = \sum_{i=1}^{k} score(q_i, d)$$

a ⋯ d_1, 0.2 | d_3, 0.1 | d_5, 0.5

b ⋯ d_5, 0.3 | d_7, 0.2

- **Main memory** proportional to number of accumulators

- Top-k result determined at the end by sorting accumulators

Term-at-a-Time

- Processes posting lists for query terms ⟨ $q_1,\ldots,q_m$ ⟩ **one at a time**
- Maintains an **accumulator for each document** seen; after processing the first **k** query terms this corresponds to

$$acc(d) = \sum_{i=1}^{k} score(q_i, d)$$

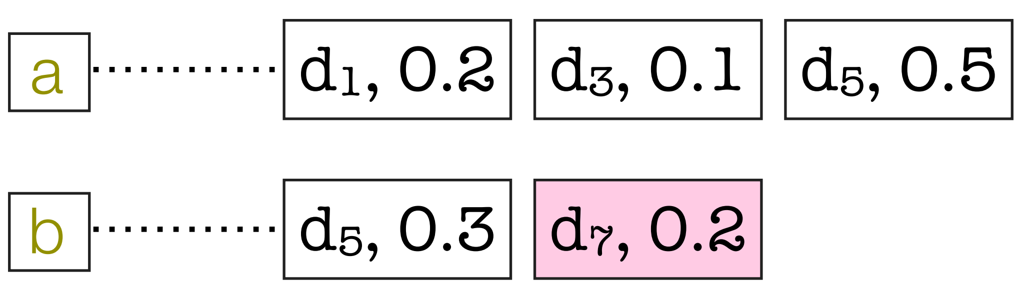
a	d_1, 0.2	d_3, 0.1	d_5, 0.5
b	d_5, 0.3	d_7, 0.2	

- **Main memory** proportional to number of accumulators
- Top-k result determined at the end by sorting accumulators

Term-at-a-Time

- Processes posting lists for query terms ⟨ $q_1,\ldots,q_m$ ⟩ **one at a time**
- Maintains an **accumulator for each document** seen; after processing the first **k** query terms this corresponds to

$$acc(d) = \sum_{i=1}^{k} score(q_i, d)$$



- **Main memory** proportional to number of accumulators
- Top-k result determined at the end by sorting accumulators

Term-at-a-Time

- Processes posting lists for query terms $\langle q_1,\ldots,q_m \rangle$ **one at a time**
- Maintains an **accumulator for each document** seen; after processing the first **k** query terms this corresponds to

$$acc(d) = \sum_{i=1}^{k} score(q_i, d)$$

a ··········· [d_1, 0.2] [d_3, 0.1] [d_5, 0.5]

b ··········· [d_5, 0.3] [d_7, 0.2]

- **Main memory** proportional to number of accumulators
- Top-k result determined at the end by sorting accumulators

Document-at-a-Time

- Processes posting lists for query terms ⟨ $q_1, \ldots, q_m$ ⟩ **all at once**
- Sees **the same document in all posting lists at the same time**, determines score, and decides whether it belongs into top-k

a	d_1, 0.2	d_3, 0.1	d_5, 0.5
b	d_5, 0.3	d_7, 0.2	

- **Main memory** proportional to **k** or number of results
- **Skipping** aids conjunctive queries (all query terms required) and can be leveraged for top-**k** queries (WAND)

Document-at-a-Time

- Processes posting lists for query terms ⟨ $q_1,\ldots,q_m$ ⟩ **all at once**

- Sees **the same document in all posting lists at the same time**, determines score, and decides whether it belongs into top-k

a	d_1, 0.2	d_3, 0.1	d_5, 0.5
b	d_5, 0.3	d_7, 0.2	

- **Main memory** proportional to **k** or number of results

- **Skipping** aids conjunctive queries (all query terms required) and can be leveraged for top-**k** queries (WAND)

Document-at-a-Time

- Processes posting lists for query terms $\langle q_1,\ldots,q_m \rangle$ **all at once**
- Sees **the same document in all posting lists at the same time**, determines score, and decides whether it belongs into top-k

a	d_1, 0.2	d_3, 0.1	d_5, 0.5
b	d_5, 0.3	d_7, 0.2	

- **Main memory** proportional to **k** or number of results
- **Skipping** aids conjunctive queries (all query terms required) and can be leveraged for top-**k** queries (WAND)

Document-at-a-Time

- Processes posting lists for query terms ⟨ $q_1, \ldots, q_m$ ⟩ **all at once**
- Sees **the same document in all posting lists at the same time**, determines score, and decides whether it belongs into top-k

a	d_1, 0.2	d_3, 0.1	d_5, 0.5
b	d_5, 0.3	d_7, 0.2	

- **Main memory** proportional to **k** or number of results
- **Skipping** aids conjunctive queries (all query terms required) and can be leveraged for top-**k** queries (WAND)

Document-at-a-Time

- Processes posting lists for query terms ⟨ $q_1,\ldots,q_m$ ⟩ **all at once**
- Sees **the same document in all posting lists at the same time**, determines score, and decides whether it belongs into top-k

a	d_1, 0.2	d_3, 0.1	d_5, 0.5
b	d_5, 0.3	d_7, 0.2	

- **Main memory** proportional to **k** or number of results
- **Skipping** aids conjunctive queries (all query terms required) and can be leveraged for top-**k** queries (WAND)

Document-at-a-Time

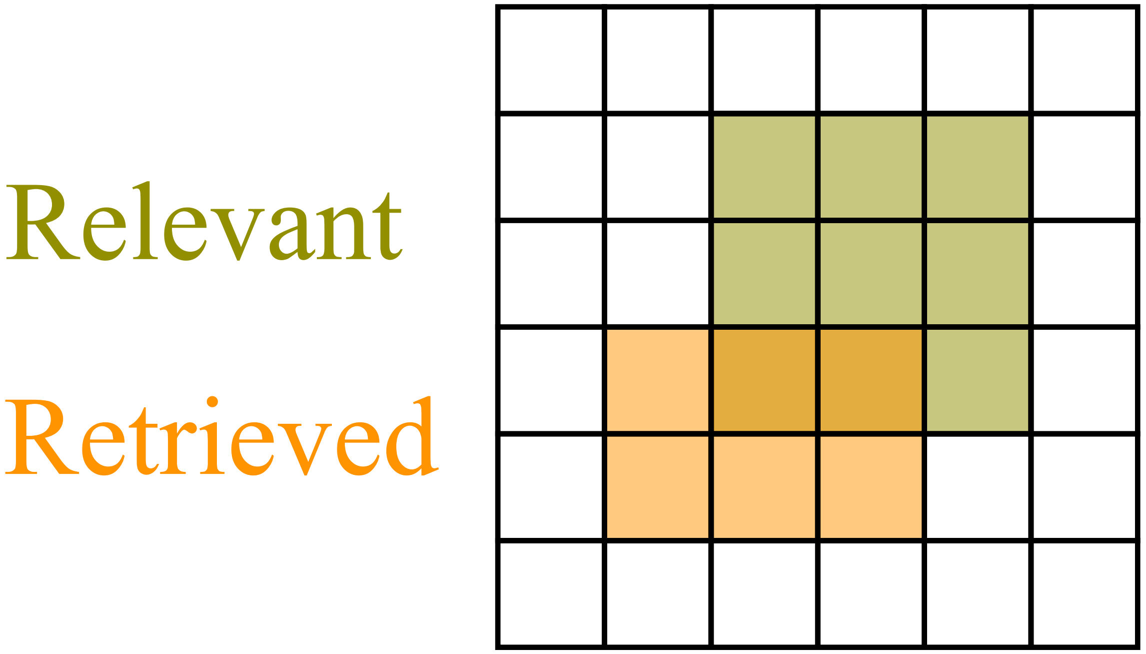
- Processes posting lists for query terms ⟨ $q_1,\ldots,q_m$ ⟩ **all at once**
- Sees **the same document in all posting lists at the same time**, determines score, and decides whether it belongs into top-k

a	d_1, 0.2	d_3, 0.1	d_5, 0.5
b	d_5, 0.3	d_7, 0.2	

- **Main memory** proportional to **k** or number of results
- **Skipping** aids conjunctive queries (all query terms required) and can be leveraged for top-**k** queries (WAND)

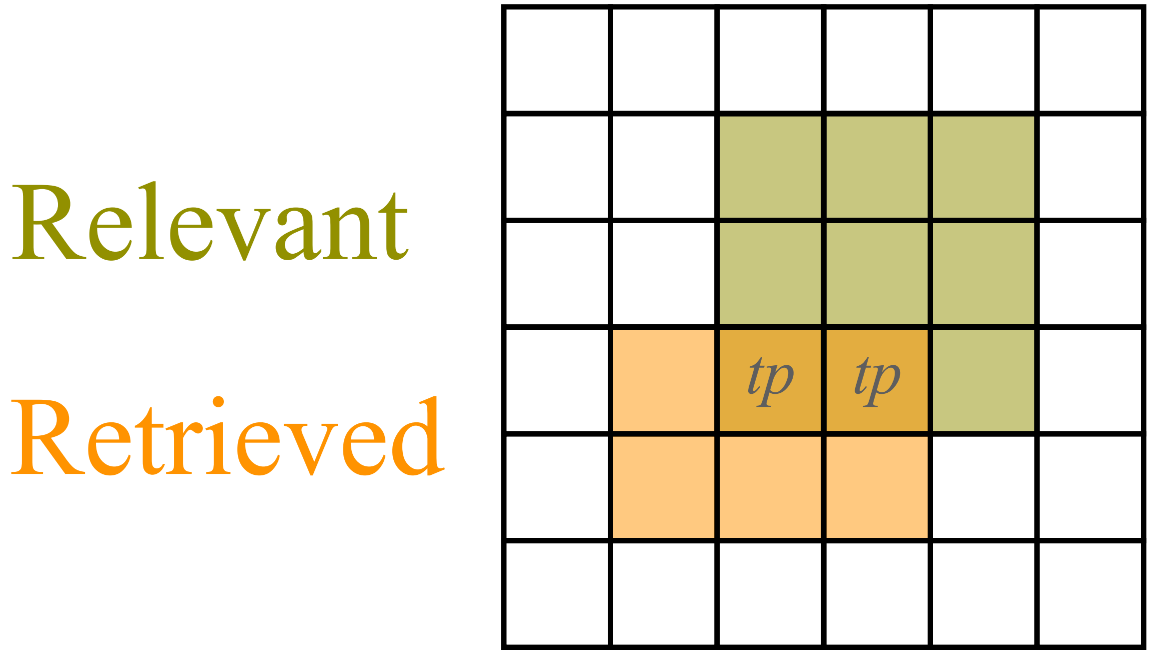
0.6. Effectiveness Measures

- We can classify documents for a given query as
  - **true positives** (tp) returned and relevant
  - **false positives** (fp) returned and irrelevant
  - **true negatives** (tn) not returned and irrelevant
  - **false negatives** (fn) not returned but relevant



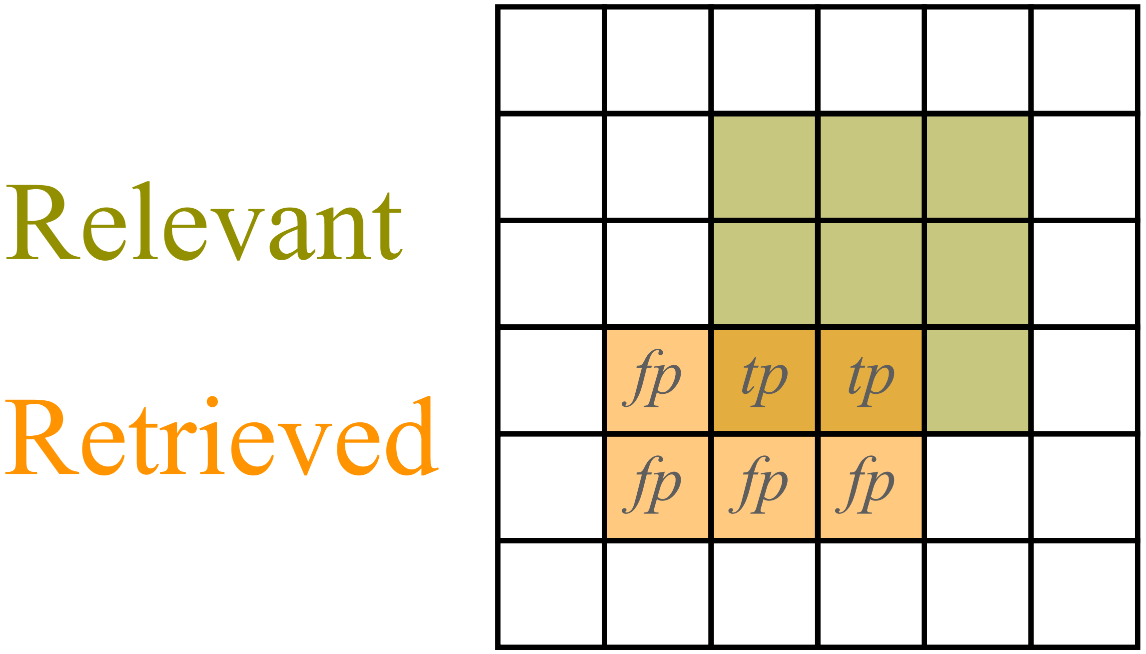
0.6. Effectiveness Measures

- We can classify documents for a given query as
  - **true positives** (tp) returned and relevant
  - **false positives** (fp) returned and irrelevant
  - **true negatives** (tn) not returned and irrelevant
  - **false negatives** (fn) not returned but relevant



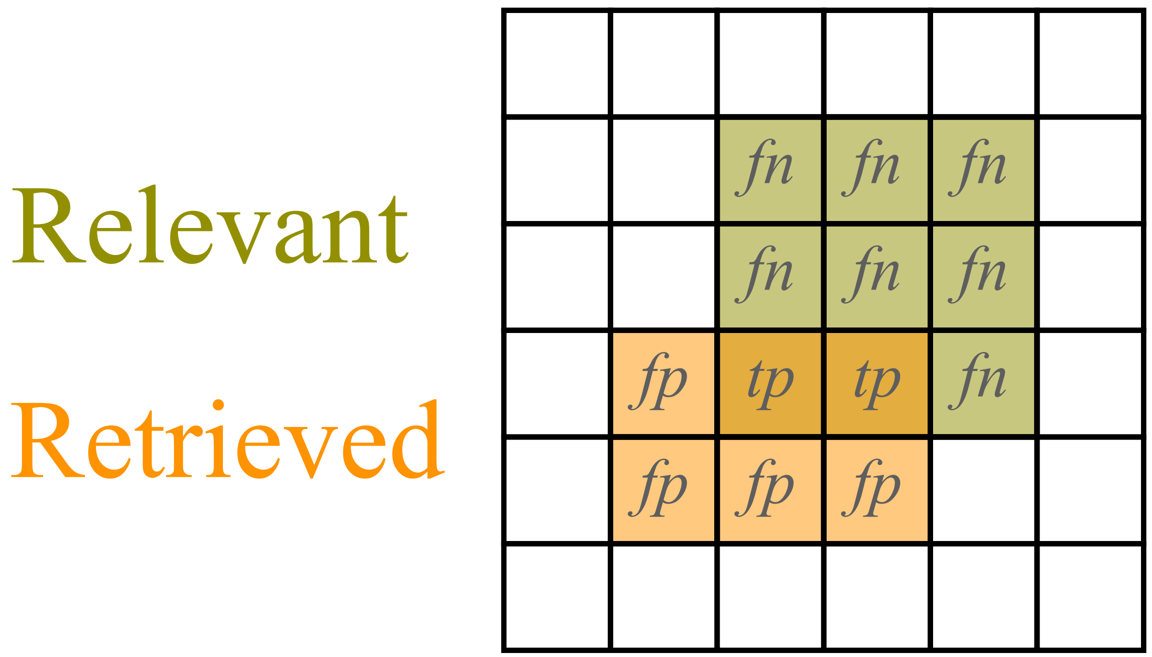
0.6. Effectiveness Measures

- We can classify documents for a given query as
  - **true positives** (tp) returned and relevant
  - **false positives** (fp) returned and irrelevant
  - **true negatives** (tn) not returned and irrelevant
  - **false negatives** (fn) not returned but relevant



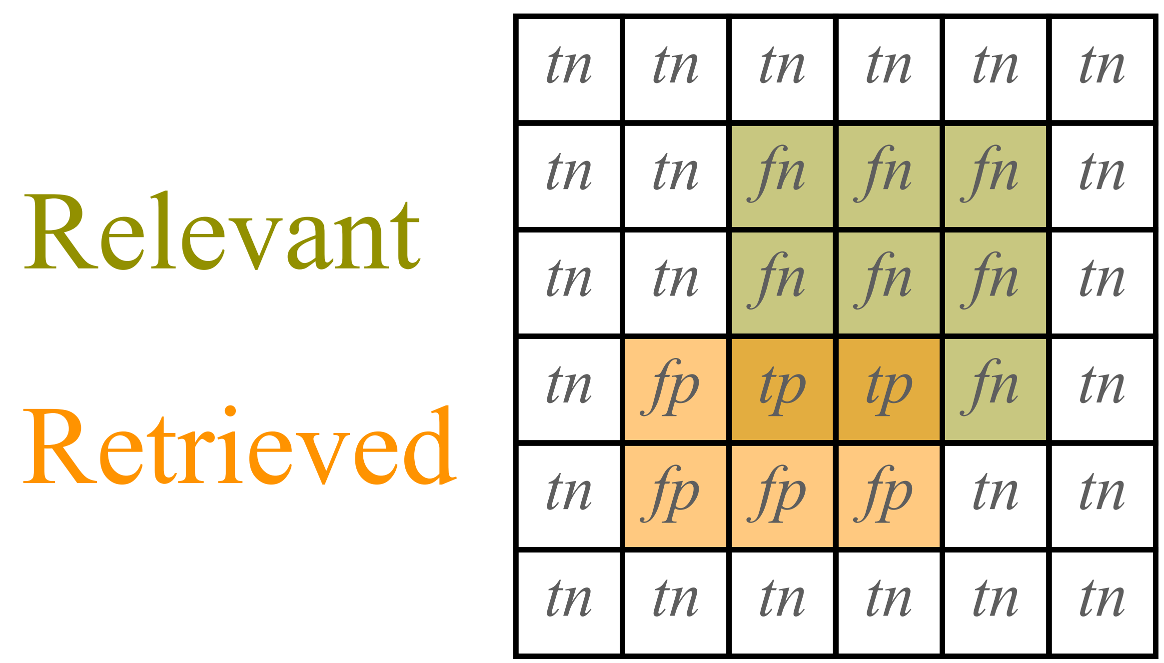
0.6. Effectiveness Measures

- We can classify documents for a given query as
  - **true positives** (tp) returned and relevant
  - **false positives** (fp) returned and irrelevant
  - **true negatives** (tn) not returned and irrelevant
  - **false negatives** (fn) not returned but relevant



0.6. Effectiveness Measures

- We can classify documents for a given query as
  - **true positives** (tp) — returned and relevant
  - **false positives** (fp) — returned and irrelevant
  - **true negatives** (tn) — not returned and irrelevant
  - **false negatives** (fn) — not returned but relevant



Precision, Recall, and F1

- **Precision** measures the ability to return **only relevant results**

$$P = \frac{\#tp}{\#tp + \#fp}$$

- **Recall** measures the ability to return **all relevant results**

$$R = \frac{\#tp}{\#tp + \#fn}$$

- **F1 score** is the harmonic mean of precision and recall

$$F_1 = 2\,\frac{P \cdot R}{P + R}$$

Normalized Discounted Cumulative Gain

- Discounted Cumulative Gain (nDCG) considers
  - **graded relevance judgments** (e.g., **2**:relevant, **1**:marginal, **0**:irrelevant)
  - **position bias** (i.e., relevant results close to the top are preferred)
- Considering top-**k** result with **R(q,m)** as grade of **m**-th document

$$DCG(q,k) = \sum_{m=1}^{k} \frac{2^{R(q,m)} - 1}{\log(1+m)}$$

- **Normalized DCG** (nDCG) obtained through normalization with idealized DCG (iDCG) of fictitious optimal top-**k** result

$$nDCG(q,k) = \frac{DCG(q,k)}{iDCG(q,k)}$$

Questions?