
Non-Standard-Datenbanken

Volltextindizierung und Information Retrieval

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

take a smile.

Non-Standard-Datenbanken

Volltextindizierung und Information Retrieval

(:)

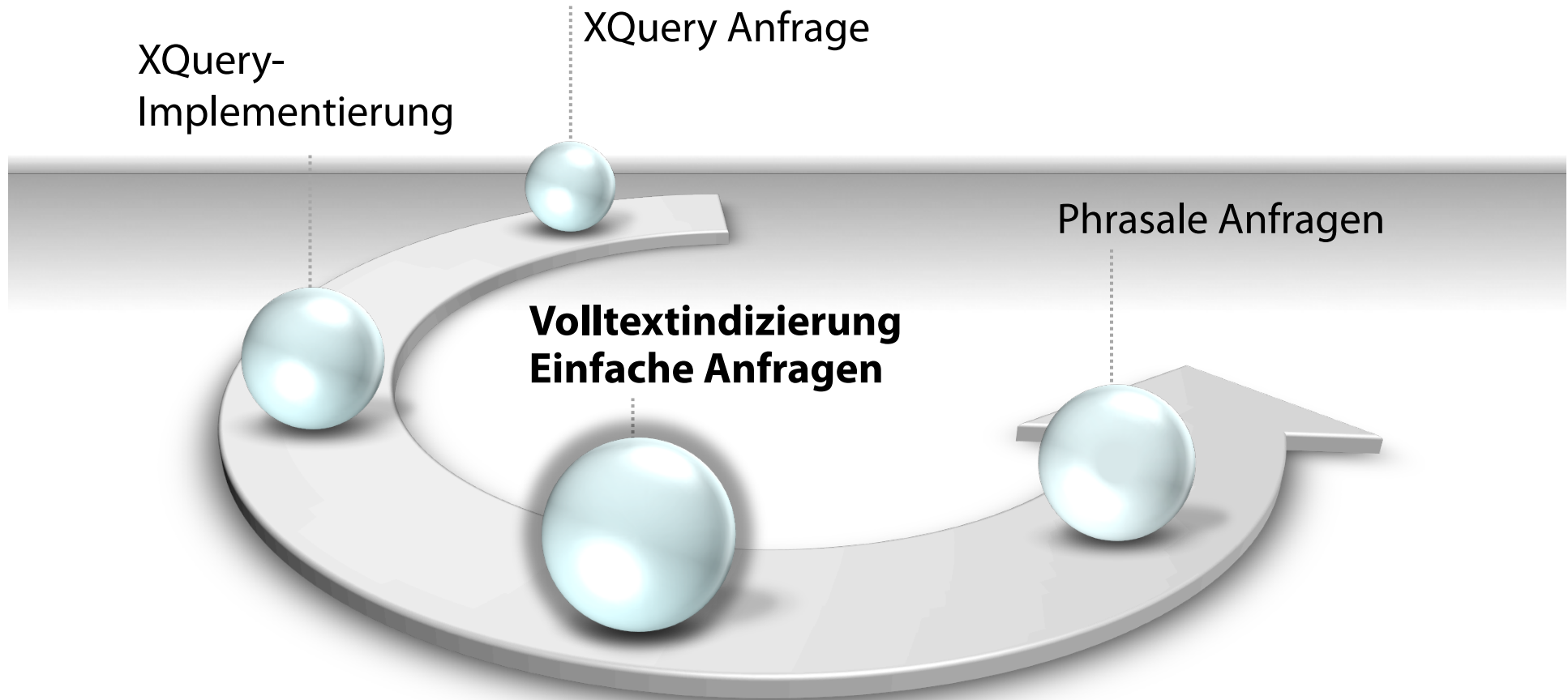
(:)

(:)

(:)

Non-Standard-Datenbanken

Von semistrukturierten Datenbanken zur Volltextsuche



Motivation

for $\$b$ in $//book$
where some $\$p$ in $\$b//para$ satisfies
contains($\$p$, "sailing") and
contains($\$p$, "windsurfing")
return $\$b/title$

Text als unstrukturierte Daten

- Welche Stücke von Shakespeare enthalten die Worte ***Brutus UND Caesar*** aber ***NICHT Calpurnia***?
- Verwendung von `grep` um alle Stücke von Shakespeare mit ***Brutus*** und ***Caesar*** zu finden, um dann Zeilen mit ***Calpurnia*** zu entfernen?
 - Langsam (für große Korpora)
 - ***NICHT Calpurnia*** ist keinesfalls einfach zu behandeln
 - Andere Operationen (z.B. Finde das Wort ***romans*** in der Nähe von ***countrymen***) sind mit `grep` nicht umsetzbar
 - Bewertung von Ergebnissen gewünscht
 - Kommt später

Term-Dokument-Inzidenzmatrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

***Brutus UND Caesar aber NICHT
Calpurnia***

*1 falls Dokument
Wort enthält, sonst 0*

Inzidenzvektoren

- Wir haben einen 0/1-Vektor für jeden Term
- Zur Beantwortung der Anfrage: Nehme die Vektoren von ***Brutus***, ***Caesar*** and ***Calpurnia*** (komplementiert) ➔ bitweises *UND*.
- $110100 \text{ UND } 110111 \text{ UND } 101111 = 100100.$

***Brutus UND Caesar aber NICHT
Calpurnia***

Antworten im Kontext

Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,
When Antony found Julius **Caesar** dead,
He cried almost to roaring; and he wept
When at Philippi he found **Brutus** slain.

*Lineare Suche des Teilausschnitts
könnte schon zu langsam sein
→ Positionen auch abspeichern*

Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius **Caesar** I was killed i' the
Capitol; **Brutus** killed me.

Größere Korpora

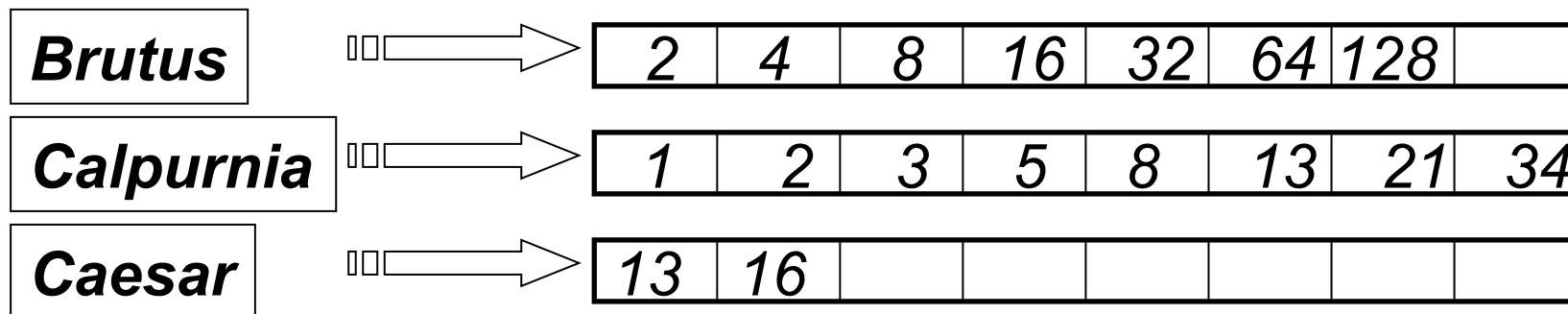
- Betrachte $N = 1\text{M}$ Dokumente, jedes mit ca. 1K Termen
- Das ergibt bei 6 Bytes/Term einschl. Leerzeichen und Interpunktionszeichen
 - 6GB Daten für alle Dokumente
- Sagen wir, es gibt darunter $m = 500\text{K}$ unterschiedliche Terme

Inzidenzmatrix viel zu groß

- 500K x 1M Matrix hat halbe Billion 0'en und 1'en
- Aber nicht mehr als eine Milliarde 1'en
 - Matrix ist extrem dünn besetzt
- Können wir eine bessere Repräsentation wählen?
 - Wir brauchen nur die 1-Positionen zu speichern

Invertierter Index

- Für jeden Term T , müssen wir eine Liste aller Dokumente, die T enthalten, speichern
- Sollen wir ein Feld oder eine Liste verwenden?

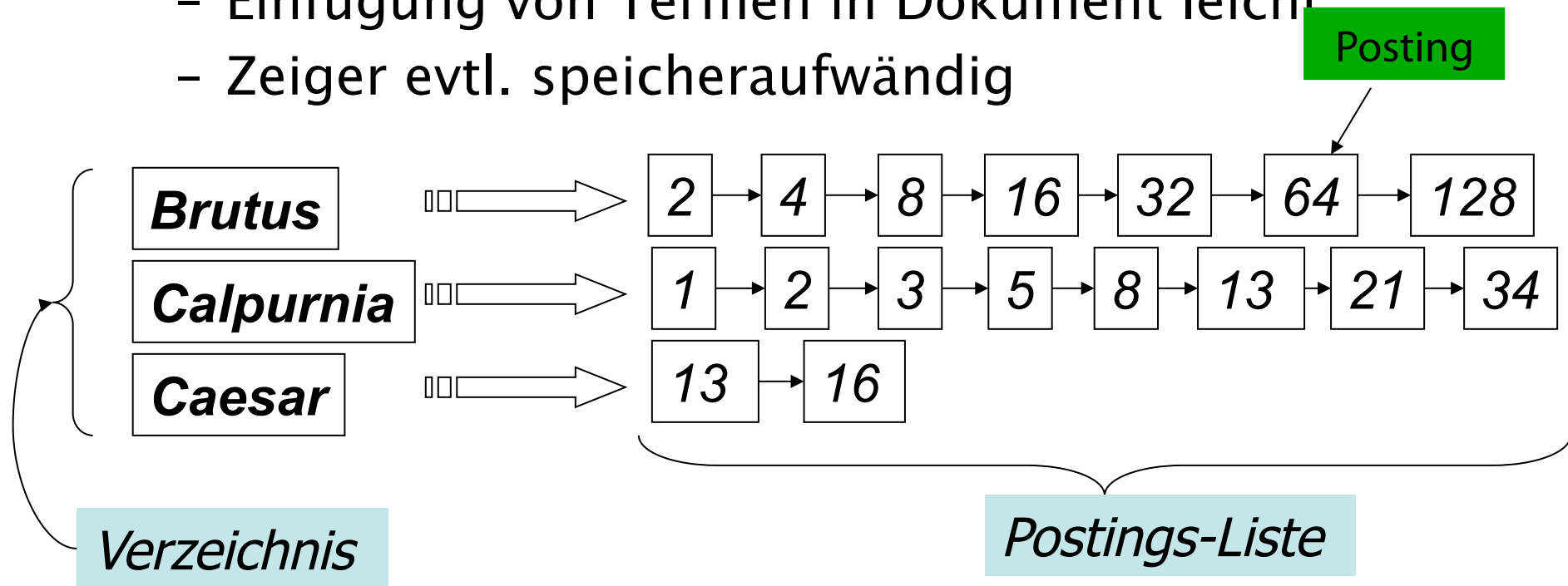


*Was machen wir, wenn das Wort **Caesar** zu Dokument 14 hinzugefügt wird?*

Luhn, H. P., Auto-encoding of documents for information retrieval systems. In M. Boaz, *Modern Trends in Documentation* (pp. 45-58). London: Pergamon Press, 1959

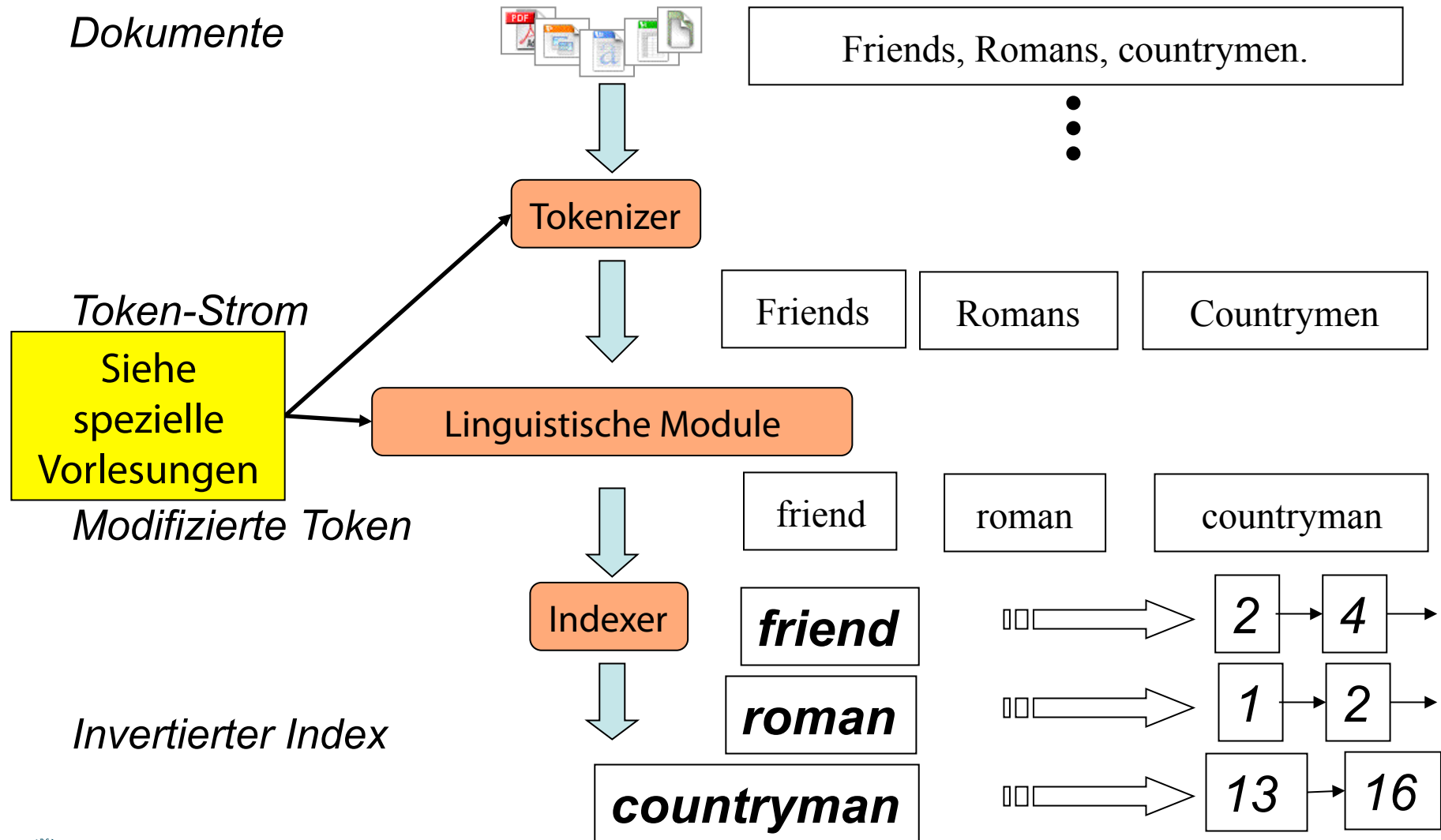
Invertierter Index

- Verkettete Listen i.a. bevorzugt
 - Dynamische Speicherallokation
 - Einfügung von Termen in Dokument leicht
 - Zeiger evtl. speicheraufwändig



Sortiert nach docID (später mehr dazu)

Konstruktion des invertierten Index



Indexierungsschritte

- Geg.: Sequenz von Paaren (Token, DocID).

Doc 1

I did enact Julius
Caesar I was killed
i' the Capitol;
Brutus killed me.

Doc 2

So let it be with
Caesar. The noble
Brutus hath told you
Caesar was ambitious



Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Indizierungsschritte

Sortierung nach Term

Hoher Aufwand

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2



Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Indizierungsschritte

- Mehrfacheinträge für Terme aus einem Dokument werden verschmolzen
- Anzahlinformation wird hinzugefügt

Warum Anzahl?
Bewertung von Antworten.
Wird später behandelt.

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



Term	Doc #	Term freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Umsetzung in SQL?

• SQL:

TermDoc

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2



Term	Doc #	Term freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Das Ergebnis wird in eine Verzeichnis- und eine Postings-Tabelle unterteilt

Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1



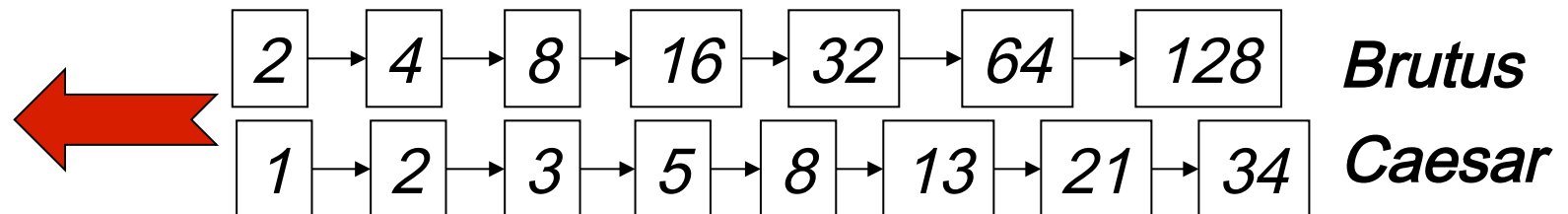
Term	N docs	Coll freq	Zeiger
ambitious	1	1	→
be	1	1	→
brutus	2	2	→
capitol	1	1	→
caesar	2	3	→
did	1	1	→
enact	1	1	→
hath	1	1	→
I	1	2	→
i'	1	1	→
it	1	1	→
julius	1	1	→
killed	1	2	→
let	1	1	→
me	1	1	→
noble	1	1	→
so	1	1	→
the	2	2	→
told	1	1	→
you	1	1	→
was	2	2	→
with	1	1	→

Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1
2	1

Warum N docs und Coll freq?
Bewertung von Antworten.
Wird später behandelt.

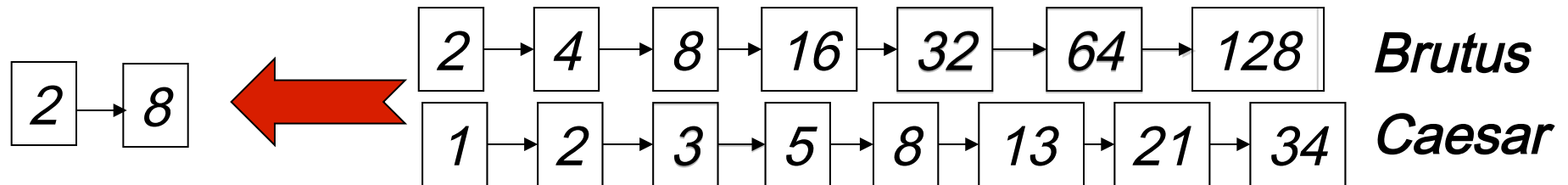
Anfrageverarbeitung: UND

- Betrachten wir folgende Anfrage:
Brutus UND Caesar
 - Suche *Brutus* im Verzeichnis
 - Hole die zugeordnete Postings-Liste
 - Suche *Caesar* im Verzeichnis
 - Hole die Postings-Liste
 - “Verschmelze” die Listen



Die Verschmelzung

- Gehe synchron durch die Postings-Listen



Falls die Listen die Länge x und y haben, dauert die Verschmelzung $O(x+y)$ Schritte.

Notwendig: Postings nach docID sortiert

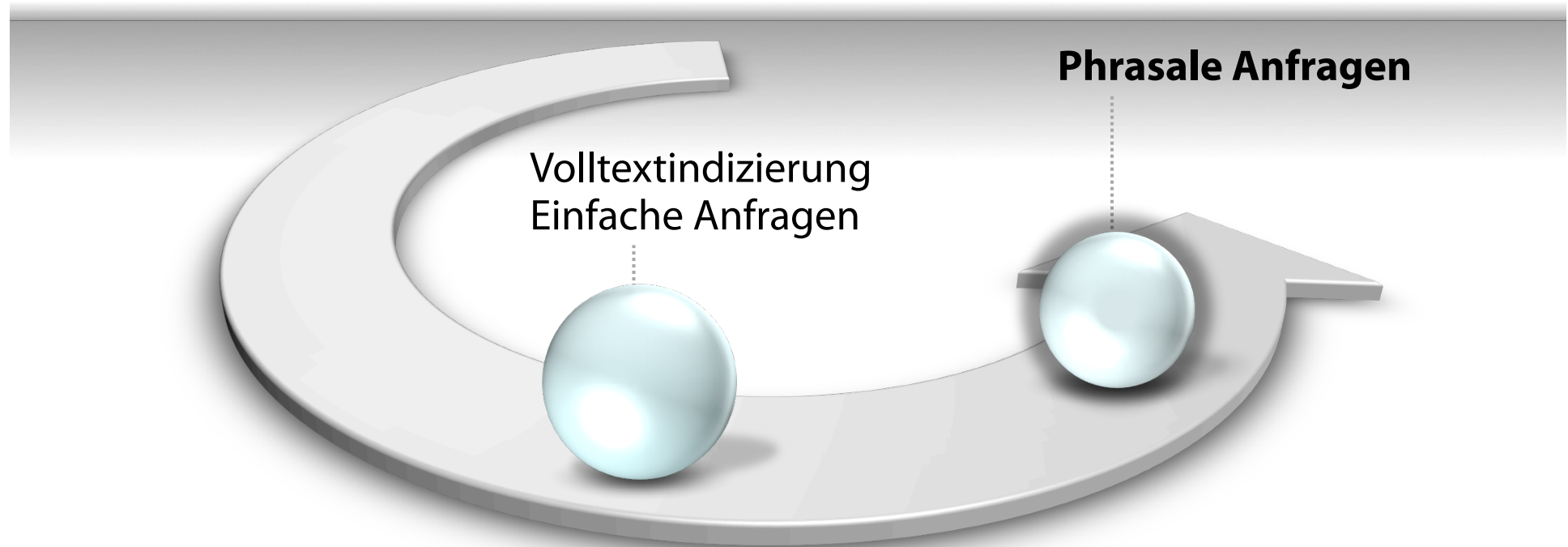
Knuth, D. E. Kap. 6.5. "Retrieval on Secondary Keys". The Art of Computer Programming, Reading, Massachusetts: Addison-Wesley, **1973**

Andere Methode: Signature Files z.B. mit Bloom Filter (hier nicht vertieft)

Bloom, Burton H., Space/Time Trade-offs in Hash Coding with Allowable Errors, Communications of the ACM 13 (7), 422–426, 1970

Non-Standard-Datenbanken

Von semistrukturierten Datenbanken zur Volltextsuche



Phrasale Anfragen

- Gesucht sind Antworten auf “*stanford university*” – als eine Phrase
- Der Satz “*I went to university at Stanford*” stellt keinen Treffer dar
 - Das Konzept der Phrasenanfrage wird von Nutzern gut verstanden und einigermaßen häufig verwendet (10% der Anfragen sind phrasal)
- Es reicht nicht, nur $\langle Term : docID \rangle$ -Einträge zu speichern


Erster Versuch: Zweiwort-Indexe

- Indexiere jedes aufeinanderfolgende Paar von Termen im Text als Phrase
- Beispieltext: “Friends, Romans, Countrymen”
- Zweiworte:
 - *friends romans*
 - *romans countrymen*
- Jedes dieser Zweiworte wird nun ein Eintrag im Verzeichnis
- Zweiwort Phrasenanfragen können nun einfach behandelt werden

Längere Phrasenanfragen

- Beispiel:
stanford university palo alto
- Behandlung als boolesche Kombination von
Zweiwortanfragen:

*stanford university UND university palo
UND palo alto*



Falsch-positive Lösungen möglich!

- Nachbetrachtung der gefundenen
Dokumente notwendig

Denkaufgabe:

Können wir den nachträglichen
Dokumentenabgleich zur Vermeidung von
falsch-positiven Ergebnissen vermeiden?

Positionsbezogene Indexe

- Speichere für jeden Term folgende Einträge:
<Anzahl der Dokumente, die Term enthalten;
Doc1: Position1, Position2 ... ;
Doc2: Position1, Position2 ... ;
...>

Beispiel: Positionsbezogener Index

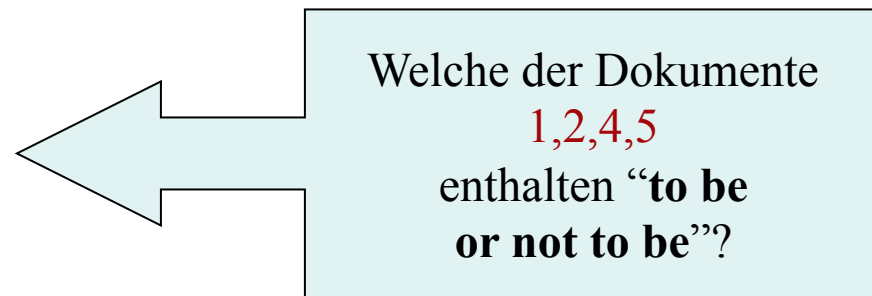
<be: 993427;

1: 7, 18, 33, 72, 86, 231;

2: 3, 149;

4: 17, 191, 291, 430, 434;

5: 363, 367, ...>



- Datenkomprimierung möglich
- Allerdings erhöht sich der Speicherverbrauch substantiell

Verarbeitung einer Phrasenanfrage

- Extrahiere die Einträge im invertierten Index für ***to, be, or, not***.
- Verschmelze die Dok-Positions-Listen um alle Positionen zu finden mit ***“to be or not to be”***.
 - ***to:***
 - 2:1,17,74,222,551; 4:8,16,190,429,433; 7:13,23,191; ...
 - ***be:***
 - 1:17,19; 4:17,191,291,430,434; 5:14,19,101; ...
- Auch anwendbar für Suchen mit “In der Nähe von”-Operatoren

Denkaufgabe:

Finde alle Dokumente, die ein Wort
enthalten, das mit "mon" beginnt.

Wie realisieren?

Platzhalter-Anfragen: *

- ***mon****: Finde alle Dokumente, die ein Wort enthalten, das mit “mon” beginnt
- Verwendung eines B-Baums mit Eintragungen in lexikographischer Ordnung
- Finde alle Worte ***w***, so dass ***mon* ≤ *w* < *moo***

Denkaufgabe:

Finde alle Dokumente, die ein Wort
enthalten, das mit "mon" endet.

Wie realisieren?

Platzhalter-Anfragen

- ****mon***: Finde Worte, die mit mon enden
- Mögliche Lösung:
 - Erstelle B-Baum für Terme rückwärts geschrieben
 - Realisierung der Anfrage als Bereichsanfrage:
nom* ≤ *w* < *non.

Denkaufgabe:

Wie können wir alle Terme (und damit Dokumente) finden, die auf **pro*cent** passen?

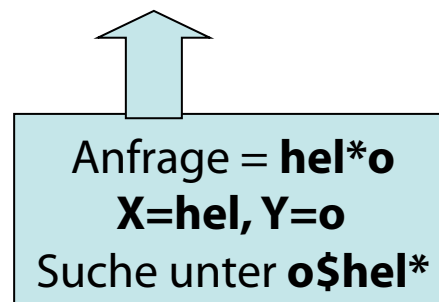
Was machen wir bei **se*ate AND fil*er**?

B-Bäume für *'ne am Ende der Anfrage

- Bei Platzhaltern in der Mitte viele Konjunkte in der resultierenden Anfrage
- Was machen wir bei multiplen Platzhaltern?
- Neuer Ansatz:
Transformiere Anfrage, so dass Platzhalter am Ende der Anfrage auftreten
→ “Permuterm”-Index

Permuterm-Index

- Für Term *hello* erstelle Indexeinträge:
 - *hello\$, ello\$h, llo\$he, lo\$hel, o\$hell, \$hello*Wobei \$ ein spezielles Symbol ist
- Behandlung von Anfragen:
 - X suche unter $X\$$ X^* suche unter $\$X^*$
 - $*X$ suche unter $X\* $*X^*$ suche unter X^*
 - X^*Y suche unter $Y\$X^*$




Herbert Marvin Ohlman
Around 1957

Denkaufgabe:

Wie behandeln wir $X*Y*Z$?

Permuterm-Anfrageverarbeitung

- Rotiere Anfrageplatzhalter nach rechts
- Verwende B-Baum-Zugriff wie bekannt
- *Permuterm-Problem: \approx Vervierfachung quadruples der zu speichernden Daten im "Lexikon"*



Empirisches Resultat für das Englische

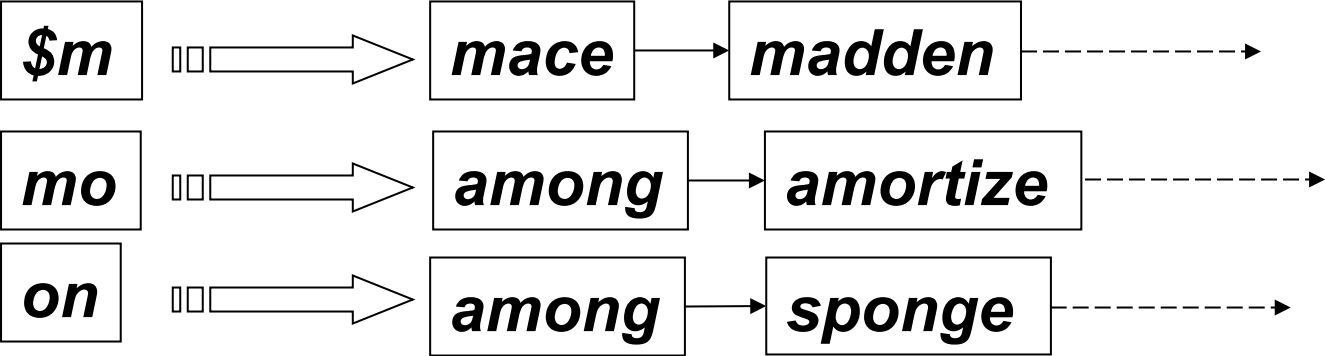
Bigramm-Indexe

- Bestimme alle n-Gramme (Sequenz von n Zeichen), die in den Termen vorkommen
- Beispieltext “***April is the cruelest month***”
Wir erhalten folgende 2-Gramme (*Bigramme*)

*\$a,ap,pr,ri,il,l\$, \$i,is,s\$, \$t,th,he,e\$, \$c,cr,ru,
ue,el,le,es,st,t\$, \$m,mo,on,nt,h\$*

- \$ ist ein besonderes Abgrenzungssymbol
- Aufbau eines invertierten Index für Bigramme auf Verzeichnisterme, in denen der Index vorkommt

Beispiel: Bigramm-Index

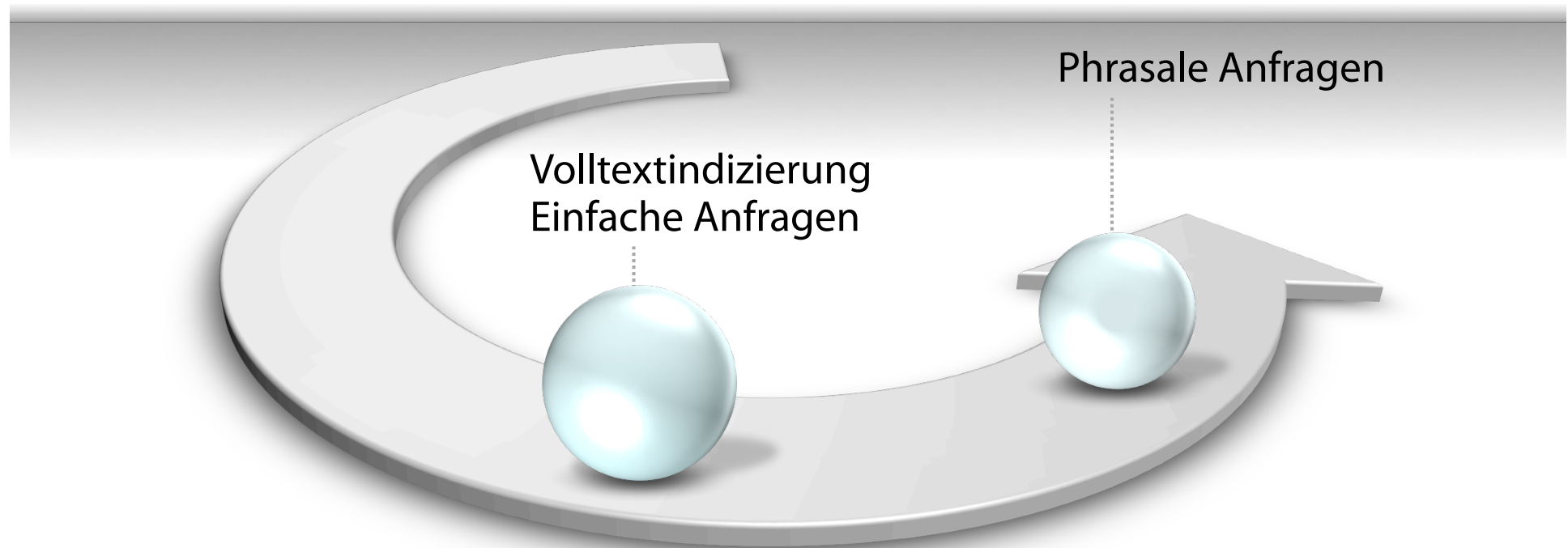


Verarbeitung von n -Gramm-Platzhaltern

- Anfrage ***mon**** kann als
 - ***\$m AND mo AND on*** formuliert werden
- Schnell und speichereffizient
- Hole Terme und gleiche die UND-Version der Platzhalteranfrage ab
- Leider kriegen wir z.B. auch ***moon*** zu fassen
- Nachverarbeitung notwendig
- Überlebende Terme führen dann über den Term-Dokument-Index zum Dokument (und ggf. zur Position darin zur Hervorhebung)

Non-Standard-Datenbanken

Volltextindizierung und phrasale Anfragen



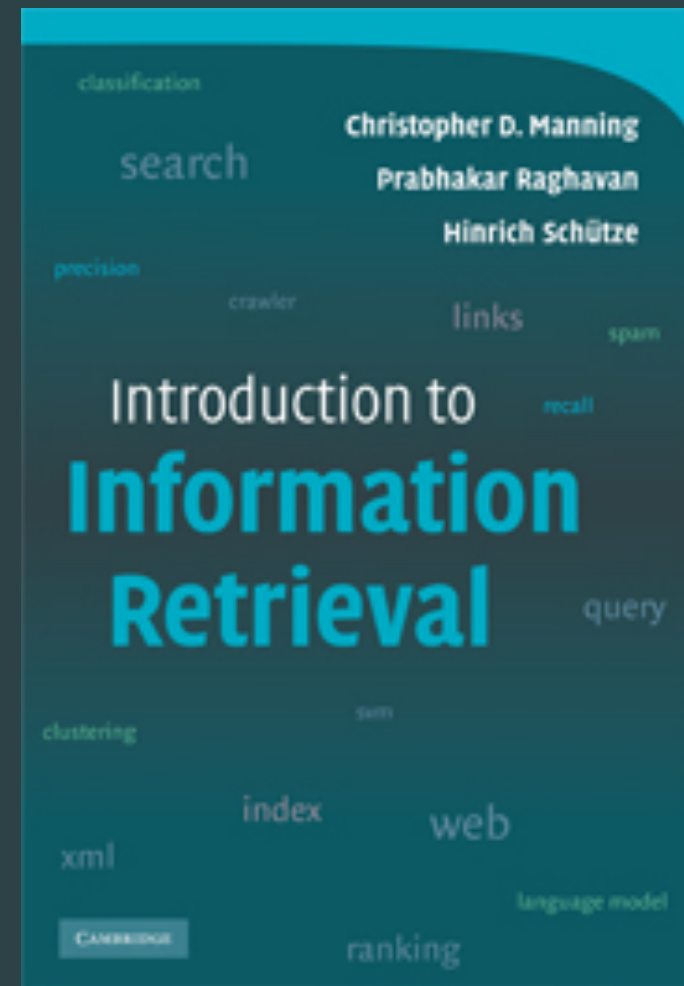
Introduction to Information Retrieval

Chr. Manning, P. Raghavan, H. Schütze

Die Präsentationen sind inspiriert durch:
<http://web.stanford.edu/class/cs276/>

Vgl. auch:

- G. Salton; E. A. Fox; H. Wu, Extended Boolean information retrieval, *Commun. ACM*, 26 (11): 10-22, 1983
- R. Baeza-Yates,; B. Ribeiro-Neto, *Modern information retrieval*, Addison-Wesley, 1999
- S. Büttcher, Ch.L.A. Clarke, G.V. Cormack,, *Information Retrieval: Implementing and Evaluating Search Engines*. Cambridge, Massachusetts: MIT Press. 2010

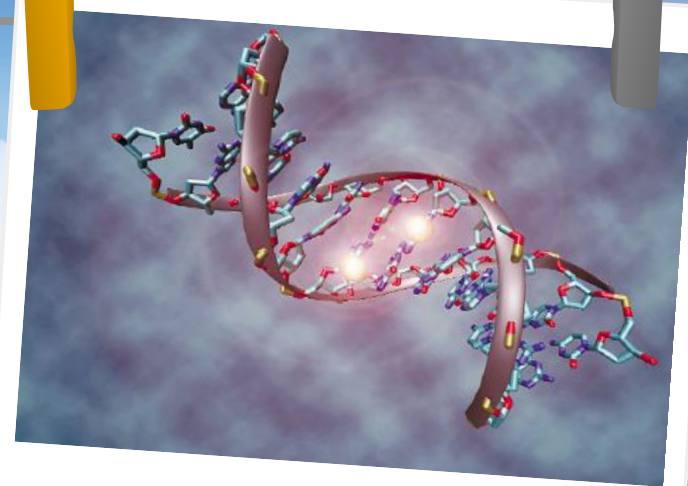




Ausblick



Temporale, räumliche und multimodale Datenbanken



Sequenzdatenbanken