Information Retrieval

Überblick

- Einführung
- Retrieval Modelle
  - Boolsches Modell
  - Vektorraummodell
  - Latent Semantic Indexing
- Evaluation
- Invertierte Listen
- Dokument Clustering, Duplikate

Motivation

- Information Retrieval
  - Repräsentation
  - Speicherung
  - Organisation
  - Zugriff
- auf Informationseinheiten
- orientiert am Bedürfnis des Anwenders
- Charakterisierung dieses Bedürfnisses
  - kein einfaches Problem

Motivation: Beispiel

- Aufgabe:
  - Finde alle Webseiten (Dokumente) über Fußballmannschaften, die
    (1) in Sachsen-(Anhalt) beheimatet sind und
    (2) in der ersten oder zweiten Bundesliga mitspielen.
    Um relevant zu sein, muss die Seite die Tabellenplätze des Vereins in den letzten drei Jahren sowie Telefonnummer und Adresse enthalten.
- Beschreibung muss in Anfrage übersetzt werden
- meistens als Menge von Schlüsselwörtern

Motivation: Beispiel

- Google: Fußball Bundesliga Sachsen Telefon Adresse
- Ergebnisse ungefähr 541 (0,17 Sekunden)
  - radio SAW - Superhits für **Sachsen**-Anhalt**...** Werdet der größte **Fußball**-Weise in **Sachsen**-Anhalt! Die neue **Fußball Bundesliga**-Saison ist im vollen Gange und radio SAW ist wieder mittendrin! **...**
  - Autobus Sippel GmbH**...** mit einem neuen Bus zu den Auswärtsspielen der **Fußball**-**Bundesliga** reisen. **...** Jäger unter **Telefon**: 06122-9124-0 oder per mail unter folgender
  - < sz-online | **sachsen** im netz >**...** Als umfassender Anbieter von Inhalten etwa zur **Fußball**-**Bundesliga** blickt **...** sich einen DSL-Anschluss mit fester IP-**Adresse** zuzulegen
  - Sehen Sie - Musik Musikfilme **Sachsen** Anhalt bei seekoo.de/regional **.......** Händler aus Deutschland mit **Telefon**-Nummer, **Adresse** und Homepage. **...** anstelle, gesundheit gewinnspiel, kraft, **fussball bundesliga** gewinnspiel, hinter,
  - autogramm -- Ideen, die bewegen.**...** Während der **Fußball**-**Bundesliga**-Saison stellen wir in jeder **...** Bitte Postkarte oder E-Mail (autogramm@ volkswagen.de) mit **Adresse**, **Telefon**-, **...**

- Das Finden von Informationen ist wichtig, im Gegensatz zum Finden von Daten

Informationen versus Daten

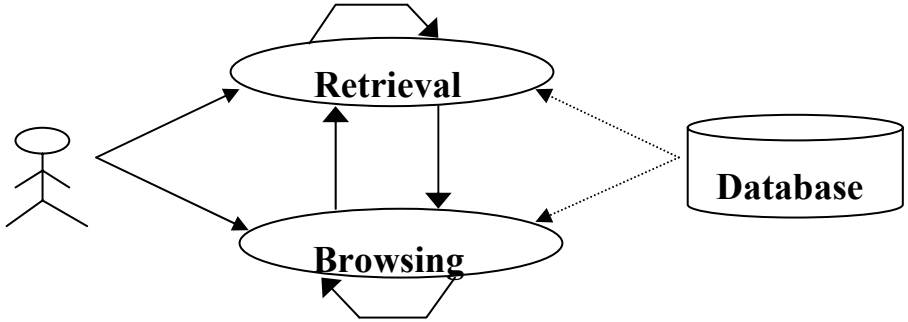
- Bereitstellen von Daten (IR)
  - welche Dokumente enthalten die Schlüsselwörter der Anfrage
  - reicht meist nicht um die Anfrage zu beantworten.
- SQL Anfrage: exakt und vollständig
- IR Anfrage: Ähnlichkeiten, kleine Fehler
  - Texte, natürliche Sprache ist nicht gut strukturiert und kann mehrdeutig sein
  - Semantik ist nicht exakt
  - Gegensatz: Datenbank-Schemata sind wohl-definiert

Informationen versus Daten

- Bereitstellen von Daten -> Datenbanken
- IR System
  - muss Dokumente „interpretieren“ und entsprechend der Relevanz ordnen
  - Syntaktische und semantische Informationen müssen aus Dokumenten extrahiert werden
  - Definition von Relevanz ist sehr wichtig
- Probleme
  - wie werden die rel. Infos extrahiert
  - wie werden die Dokumente geordnet
- Ziel
  - Anwort enthält alle relevanten Dokumente mit möglichst wenigen nicht-relevanten Dokumenten

Grundlegende Konzepte, Anwender Aufgabe

- Anwender übersetzt dem IR System Aufgabe in Anfrage (Schlüsselwörter+Bedingungen)
- Anwender mit unbestimmten, allgemeinem Interesse
  - will einfach herumstöbern
  - **stöbern** anstatt **suchen** (klare Unterscheidung)



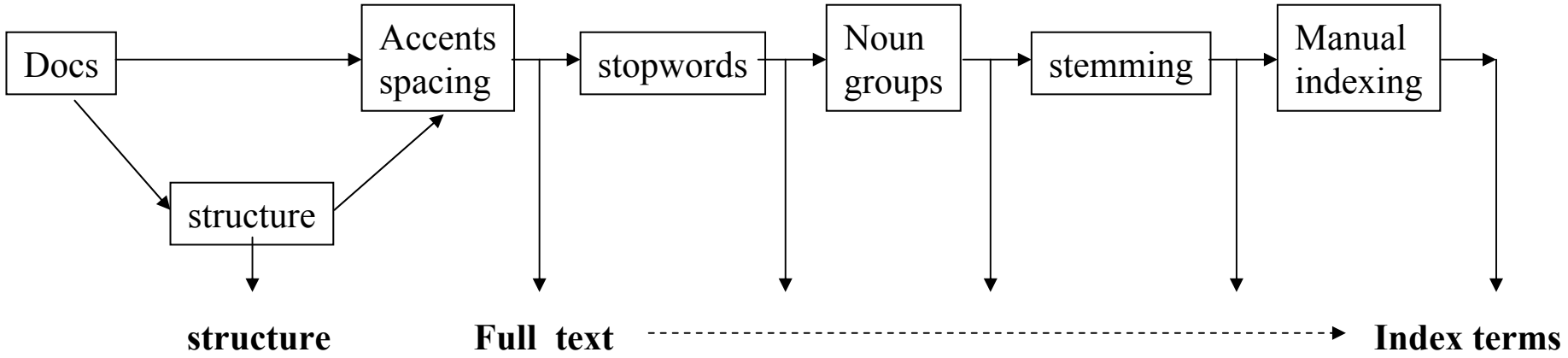
Grundlegende Konzepte, Logische Sicht auf Dokumente

- Texte repräsentiert durch eine Menge von Indexwörtern => logische Sicht
  - handverlesene Indexmenge
  - automatisch generiert
- Indexwörter = alle Wörter => Volltext
- Textoperationen
  - Stopwörter
  - Stemming
  - Substantiv-Gruppen

Dokument Vorverarbeitung

- Lexikalische Analyse
  - Behandlung von Zahlen, Bindestrichen, Punktierungen, Groß/Kleinschreibung
  - Implementierung einfacher Regeln + Liste von Ausnahmen
- Eliminierung von Stop-Wörtern
  - Wenig selektive Wörter entfernen
  - Listen
- Stemming
  - Einfache grammatische Regeln (sprachabhängig)
- Auswahl der Indexterme
  - Meist Substantive und Wortgruppen
- Konstruktion eines Thesaurus
  - Hilfe für Anwender, Anfrageerweiterung

Logische Sichten auf Dokumente



Dokument Vorverarbeitung

- Hilfreiche Seiten
- Stopwortlisten, Stemmer für verschiedene Sprachen
  http://www.unine.ch/info/clef/
- Deutsche und Englische Wörterbücher
  http://dict.tu-chemnitz.de/
- Thesauri
  - http://www.openthesaurus.de/
  - http://thesaurus.reference.com/
  - http://wordnet.princeton.edu/

Textsuche mit Indexstrukturen

- Zu einer Anfrage alle Dokumente sequentiell durchsuchen?
  - Kleine Textsammlungen (ein paar MB)
  - Sehr viele Änderungen
- Indexstrukturen
  - Mehraufwand lohnt sich erst für große Texte
  - Text ist semi-statisch
- Typen
  - Inverted Files
  - Suffix Bäume
  - Signature Files

Anfragen

- Welches Stück von Shakespeare enhält die Wörter **Brutus**, **Caesar** aber NICHT **Calpurnia**?
- Ansatz: finde alle Stücke mit Brutus und Caesar und entferne alle die Calpurnia enthalten
  - Langsam für große Sammlungen
  - NICHT Calpurnia ist nicht trivial

Term-Dokument Inzidenz Matrix

	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

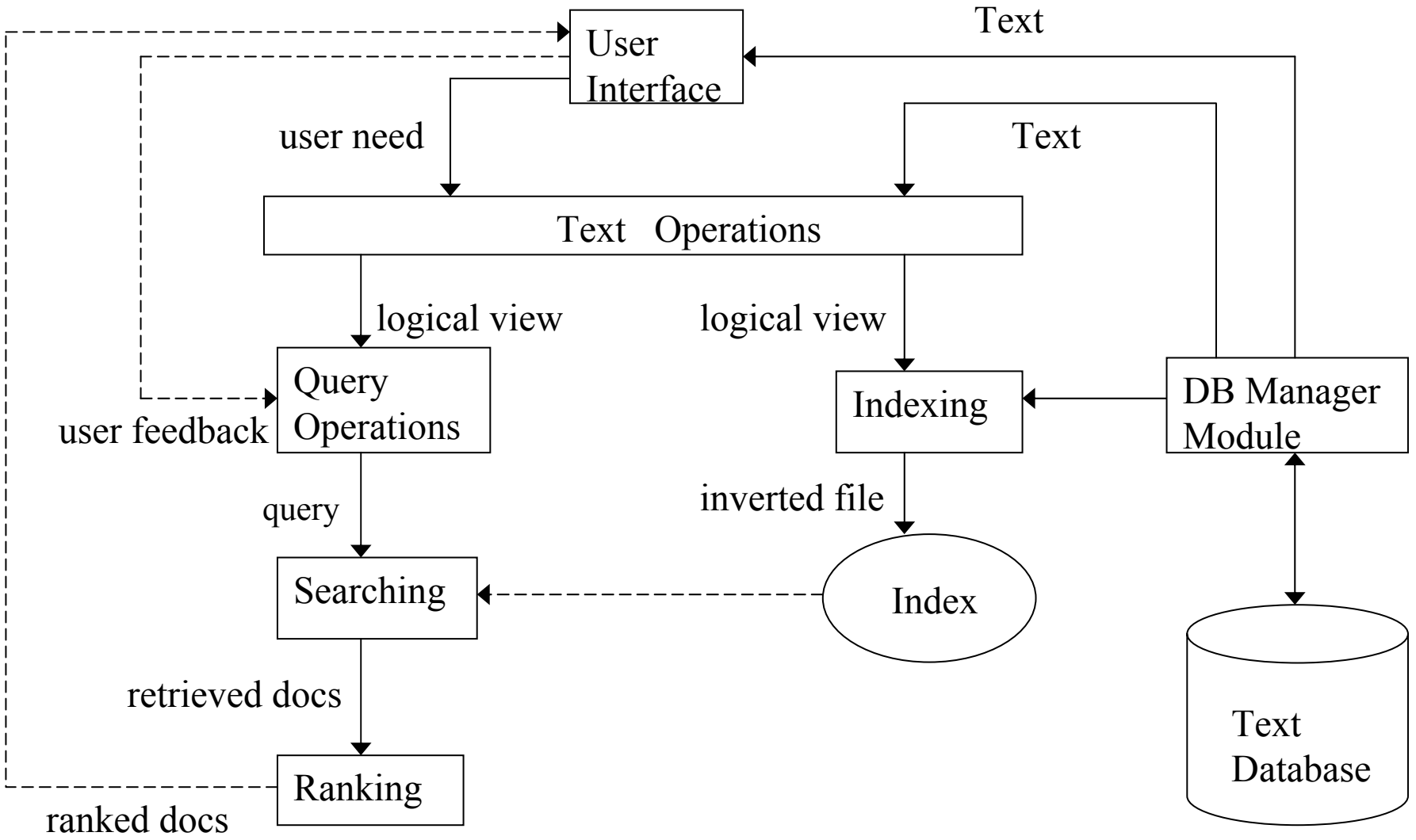
Inzidenz Vektoren

- 0/1 Vektoren für jeden Term
- Anfrage beantworten: bit weises AND der Vektoren für Brutus, Caesar und Calpurnia (komplementiert)

110100 *AND* 110111 *AND* 101111 = 100100

- Antworten:
  - Antony and Cleopatra
  - Hamlet

Suche als Prozess

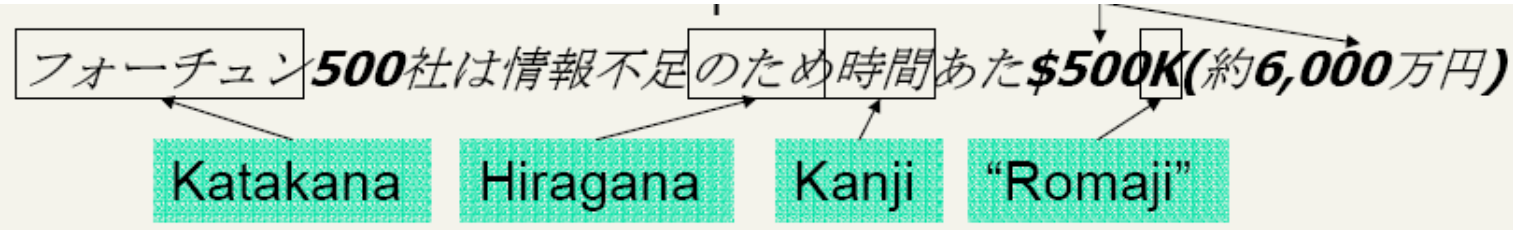


Anfrage-Formulierung

- IR Anfragesprachen erlauben Ranking
- Anfragen mit Schlüsselwörtern
  - Einzelwörter
  - Kontext Anfragen
  - Boolsche Anfragen
  - Natürliche Sprache
- Textmuster
- Strukturanfragen

Anfragen mit Schlüsselwörtern

- Text in Einheiten zerlegen
  - Eingabe: “Freunde, Römer, Landsleute ...“
  - Ausgabe: {“Freunde“,“Römer“,“Landsleute“}
- Wörter als Grundeinheit, was ist ein Wort?
- Sprachprobleme
  - Chinesisch, Japanisch: keine Leerzeichen
  - Japanisch: mehrere Alphabete gemischt



Sprachprobleme

- Normalisierung
  - "rechts-nach-links" Sprachen, (Hebräisch, Arabisch), gemischt mit "links-nach-rechts" Text (Datum, Geld, …)
  - Groß/Kleinschreibung, Mehrdeutigkeiten
    - „Morgen will ich im MIT"
    - Alles Kleinschreiben, Ausnahmen
  - Punktierung
  - Zahlen

Rechtschreibprüfung

- Erweitere Anfragen
  - z.B. um Terme innerhalb Edit-Distanz 2
  - Erweitere zur Anfragezeit

- Rechtschreibprüfung ist teuer, verlangsamt Anfrage bis zu Faktor 100
  - Wird eingesetzt, wenn fast nichts gefunden wird
  - Was, wenn Dokumente Schreibfehler enthalten ?

Soundex

- Heuristiken um Anfrage um phonetische Äquivalente zu erweitern
  - Sprachspezifisch, hauptsächlich für Namen
- Typischer Algorithmus, -> 4 Zeichensignatur
  1. Erhalte ersten Buchstaben des Wortes
  2. Ersetze
     - A E I O U H W Y zu 0
     - B F P V zu 1
     - C G J K Q S X Z zu 2
     - D T zu 3
     - L zu 4
     - M N zu 5
     - R zu 6
  3. Entferne alle aufeinanderfolgenden Duplikate
  4. Entferne alle Nullen
  5. Liefere die ersten vier Positionen (bei Bedarf mit Nullen auffüllen)

Überblick

- Einführung
- Retrieval Modelle
  - Boolsches Modell
  - Vektorraummodell
  - Latent Semantic Indexing
- Evaluation
- Invertierte Listen
- Dokument Clustering, Duplikate

Boolsches Modell

- Vorteil: einfach verständlich
- Nachteile
  - Keine Abstufung der Relevanz
  - Informationsbedürfnis schwer in logische Formel zu übersetzen
- Problem: kein teilweises Erfüllen möglich
- Folge: entweder zu wenig oder zu viele Ergebnistreffer

Vektorraum Modell (1/2)

- Dokumente werden durch Vektoren beschrieben

$$\vec{d_j} = (w_{1,j}, \ldots, w_{t,j}),\ w_{i,j} \geq 0$$

- Anfragen sind auch Vektoren

$$\vec{q} = (w_{1,q}, \ldots, w_{t,q}),\ w_{i,q} \geq 0$$

- Similarity, Ähnlichkeit
  - Cosinus des Winkels zwischen *q* und *d*, Cosine Similarity

$$sim(\vec{d_j}, \vec{q}) = \frac{\vec{d_j} \cdot \vec{q}}{\|\vec{d_j}\| \cdot \|\vec{q}\|} = \frac{\sum_{i=1}^{t} w_{i,j} \cdot w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \cdot \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

t ist die Anzahl der verschiedenen Wörter (Größe des Vokabulars)

Vektorraum Modell (2/2)

- Vorteile
  - Bessere Retrieval Ergebnisse
  - Teiltreffer möglich
  - Relevante Dokumente nach Ähnlichkeit sort.
- Nachteil: unabhängige Termgewichte
- Fazit:
  - im klassischen Rahmen schwer zu schlagen
  - sehr populär

Latent Semantic Indexing

- Texte beschreiben (latente) Konzepte mit Hilfe eine Menge von Worten (Indexterme)
- Ranking basierend auf Konzepten anstelle von Indextermen
- Idee
  - Ausgehend vom Term-Vektorraum $\mathbf{R}^t$ finde einen Unterraum $\mathbf{R}^s$ mit s << t (t ~ 500.000, s ~ 100 – 300)
  - Termvektoren, die gleiche Konzepte beschreiben, sollen auf ähnliche Punkte im Konzept-Vektorraum $\mathbf{R}^s$ abgebildet werden

Latent Semantic Indexing

- Fazit:
  - LSI (SVD Kompression) erhöht die Retrieval-Qualität,
  - auch wenn der Approximationsfehler hoch ist
  - LSI kann mit Synonymen umgehen (Baby, Kind, Sohn, Tochter, ...)
  - LSI kann mit Mehrfachbedeutungen umgehen (Bank, Ball, ...)
  - LSI funktioniert auch für allg. Dokumentsammlungen, wobei s erstaunlich niedrig ist.
  - Verbessert nicht in jede Anfrage, jedoch die meisten (Verbesserung der Durchschnittsqualität)

Überblick

- Einführung
- Retrieval Modelle
  - Boolsches Modell
  - Vektorraummodell
  - Latent Semantic Indexing
- Evaluation
- Invertierte Listen
- Dokument Clustering, Duplikate

Evaluierung von IR Systemen (1/2)

- Theoretische Evaluierung kaum möglich
- Evaluierung von IR Systemen ist experimentell
  - Zuverlässigkeit: dieselbe Untersuchung im gleichen Kontext sollte stets dieselben Ergebnissen liefern (Wiederholbarkeit)
  - Validität: die Beobachtungen sollten mit den 'tatsächlichen, Verhältnissen übereinstimmen (Gültigkeit)
  - Stochastische Experimente: Meßwerte können variieren
- Performanzmaße für Effizienz
  - Speicherplatz
  - CPU-Zeit
  - Anzahl I/O-Operationen
  - Antwortzeiten

Evaluierung von IR Systemen (2/2)

- Qulität
  - Relevanz der Antwort
- Annahmen
  - Die Systemantwort ist eine Menge von Objekten. Damit werden stärker strukturierte Antworten nicht berücksichtigt. Erweiterung auf lineare Anordnungen (Rangordnungen) leicht möglich
  - Die Relevanz eines Objekts bezüglich der Anfrage (Qualität) hängt nur von der Anfrage ab. Wechselseitige Abhängigkeiten zwischen Objekten bleiben dagegen unberücksichtigt. (wenn z. B. die Bedeutung eines bestimmten Dokumentes erst nach der Lektüre eines anderen Dokumentes erkannt wird).
- Arten von Relevanz
  - Situative Relevanz
  - Pertinenz
  - Objektive Relevanz
  - Systemrelevanz

Evaluierung von IR Systemen (2/2)

- Standpunkte und Bewertungsmaße
  - IRS lieftert eine Rangordnung von Dokumenten
  - Benutzer geht diese sequentiell durch, bis ein bestimmtes Abbruchkriterium erfüllt ist.
  - Beispiele für Abbruchkrit. und Bewertungsmaß
    - n Dokumente gesehen: # gesehene relevante Dokumente
    - n relevante Dokumente gesehen: # gesehene Dokumente
    - n nicht relevante Dokumente gesehen: # gesehene / # gesehene relevante Dokumente
    - n nicht relevante Dokumente in Folge gesehen: # gesehene / # gesehene relevante Dokumente
- Test-Referenzsammlungen
  - Anfragen plus relevante Dokumente

Recall, Precision, Fallout

- Precision
  - Anteil der relevanten Dokumente unter den gefundenen Dokumenten
- Recall
  - Anteil der gefundenen relevanten Dokumente unter allen relevanten Dokumenten
- Fallout
  - Anteil der gefundenen irrelevanten an allen irrelevanten Dokumenten der Kollektion
- Annahme
  - alle Dokumente aus A wurden vom Anwender bewertet
  - bei Top-r Rankings variiert Recall und Precision mit steigendem r

Recall, Precision

- Einzelne PR-Werte
  - Web-Retrieval
    - die meisten Nutzer schauen nur die erste Seite an
      => Prec@10 , Google: Prec@1
    - Simulation von Benutzerklassen
      => Prec@5, Prec@10, Prec@30, Prec@100
- Standpunkte
  - einzelner Benutzer
    - Zusammenfassung der eigenen Anfragen
  - Systembetreiber
    - Zusammenfassung der Anfragen aller Benutzer

Beispiel: Recall, Precision

- Gegeben eine Anfrage q
- R = {d3, d9, d25, d56, d123}
- Ranking eines IR Systems für Anfrage q

1. d123	6. d9	11. d38
2. d84	7. d511	12. d48
3. d56	8. d129	13. d250
4. d6	9. d187	14. d113
5. d8	10. d25	15. d3

Recall, Precision

- Standard Recall Stufen
  - 0%, 10%, 20%, …, 100%
- Precision Werte für mehrere Anfragen werden gemittelt
- Interpolation nach Salton
  - Ziehe von jedem PR-Punkte ein waagerechte Linie nach links
  - Ergebnis PR-Graph ist das Maximum

Probleme

- Alle relevanten Dokumente müssen bekannt sein
  - ist für große Dokumentmenge schwer zu garantieren
  - Precision-Recall sind nicht unabhängig
  - Interaktivität wird nicht berücksichtigt

Überblick

- Einführung
- Retrieval Modelle
  - Boolsches Modell
  - Vektorraummodell
  - Latent Semantic Indexing
- Evaluation
- Invertierte Listen
- Dokument Clustering, Duplikate

Textsuche mit Indexstrukturen

- Zu einer Anfrage alle Dokumente sequentiell durchsuchen?
  - Kleine Textsammlungen (ein paar MB)
  - Sehr viele Änderungen
- Indexstrukturen
  - Mehraufwand lohnt sich erst für große Texte
  - Text ist semi-statisch
- Typen
  - Inverted Files
  - Suffix Bäume
  - Signature Files

Anfragen

- Welches Stück von Shakespeare enhält die Wörter **Brutus**, **Caesar** aber NICHT **Calpurnia**?
- Ansatz: finde alle Stücke mit Brutus und Caesar und entferne alle die Calpurnia enthalten
  - Langsam für große Sammlungen
  - NICHT Calpurnia ist nicht trivial

Term-Dokument Inzidenz Matrix

	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

Inzidenz Vektoren

- 0/1 Vektoren für jeden Term
- Anfrage beantworten: bit weises AND der Vektoren für Brutus, Caesar und Calpurnia (komplementiert)

110100 *AND* 110111 *AND* 101111 = 100100

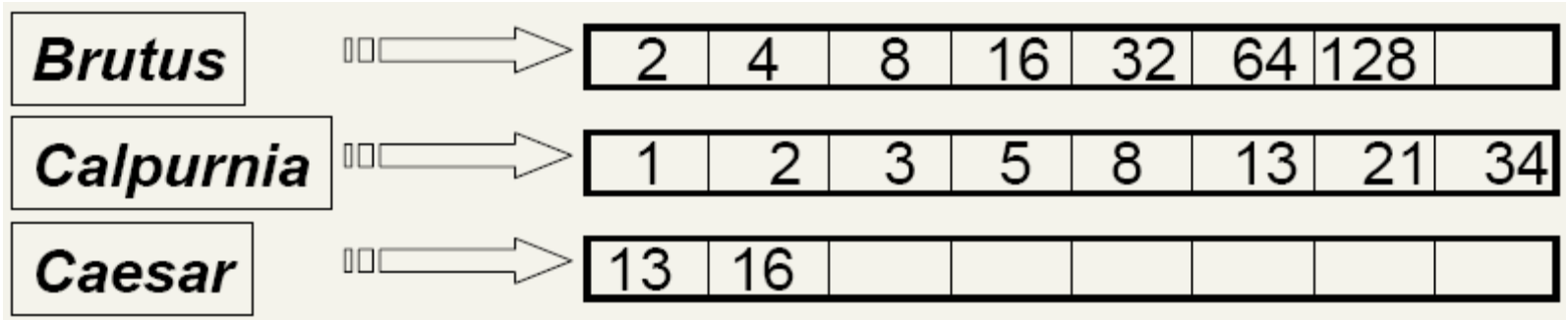
- Antworten:
  - Antony and Cleopatra
  - Hamlet

Größere Sammlungen

- Annahmen
  - N=1M Dokumente, jedes mit etwa 1K Termen
  - Avg. 6 bytes/Term mit Leerzeichen/Punktuation
    - 6GB Textdaten in den Dokumenten
  - t=500K verschiedene Terme
- Matrix kann nicht materialisiert werden
  - 500K * 1M = ½ * 10^12 Einträge
  - Nicht mehr als 10^9 Eins-Einträge, Warum?
- Besserer Ansatz: Repräsentiere nur die Einsen

Inverted Files

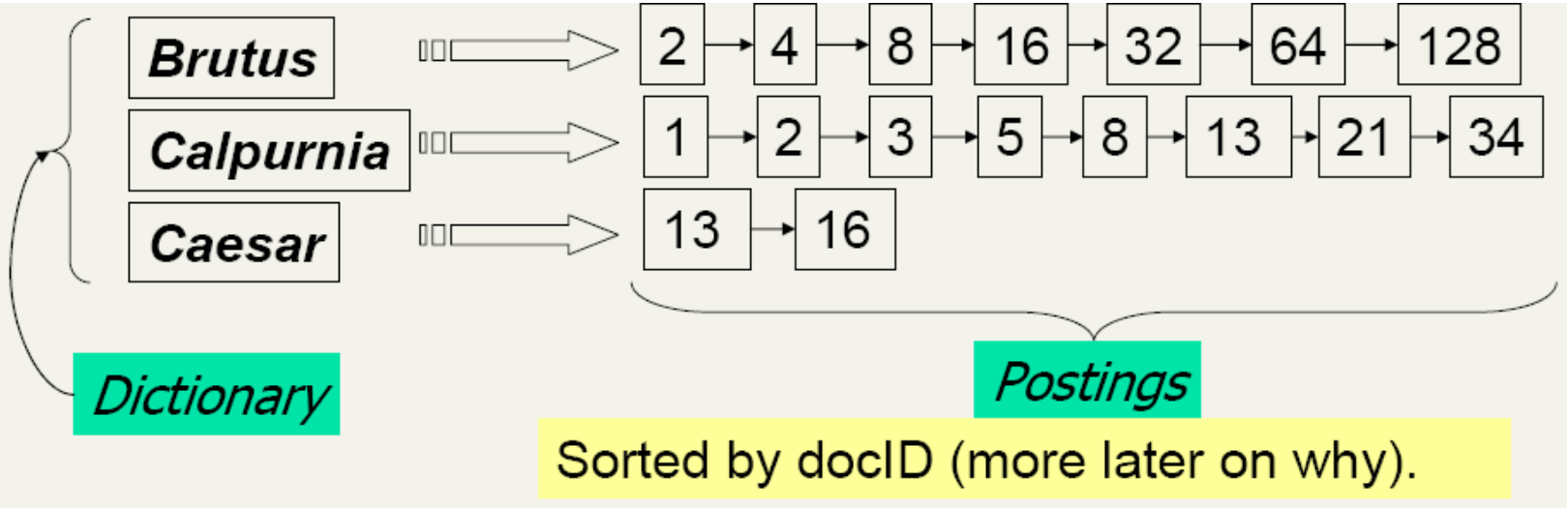
- Zu jedem Term k speichere alle Dokumente, die k enthalten
  - Felder oder Listen?



  - Was passiert wenn das Wort Caesar zu Dok. 14 hinzugefügt wird?

Inverted Files

- Listen werden meist Feldern vorgezogen
  - Dynamischer Platz
  - Einfügen von Termen in Dok. leicht
  - Mehrverbrauch an Platz für Zeiger



Index Schritte (1)

- Liste von (Termen, Dok. ID) Paaren

Dok. 1

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

Dok. 2

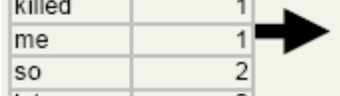
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

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

Sortieren der Terme

- Hauptschritt für Indexing

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

Mehrfach Einträge

- Mehrfach Einträge in einzelnen Dok. zusammenfassen
- Häufigkeit hinzufügen

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 #	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

Teile Ergebnis in Dictionary und Postings

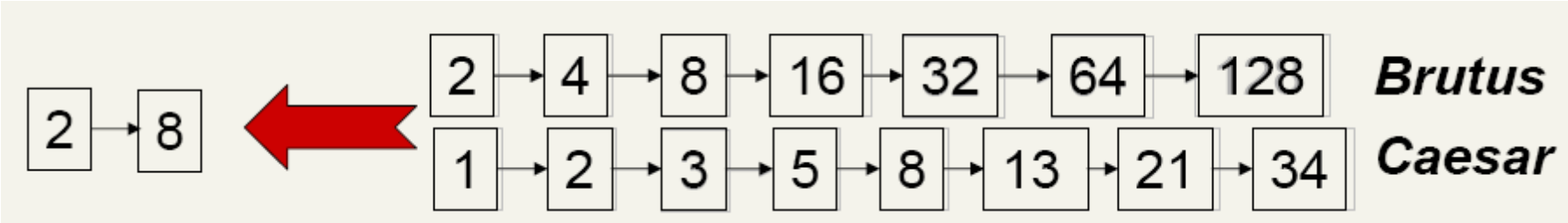
Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Term	N docs	Tot Freq
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	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

Anfrageverarbeitung

- Anfrage: Brutus AND Caesar
  - Finde Postings von Brutus und Caesar
  - Fasse beide **sortierte** Listen zusammen, durchlaufe beide Listen simultan



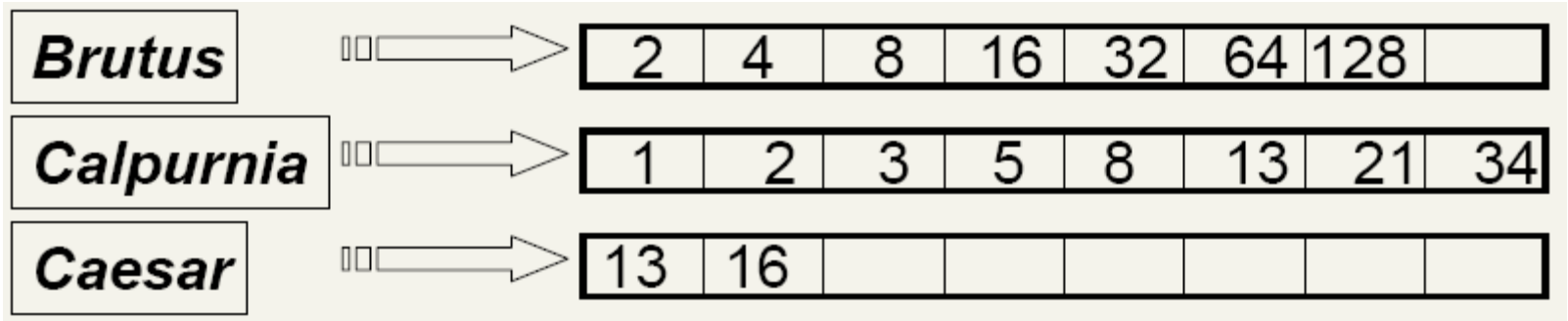
  - Wenn Listenlängen x und y sind, Schnitt in O(x+y)

Allgemeinere Anfragen

- Wie muß das Zusammenfassen geändert werden für
  - Brutus AND NOT Caesar
  - Brutus OR Caesar
  - Brutus OR NOT Caesar
- Bleibt die Laufzeit bei O(x+y)
- Was passiert bei allg. Boolschen Anfragen?
  - (Brutus or Caesar) and not (Antony or Cleopatra)
  - Kann man immer in “linearer” Zeit zusammenfassen
  - Linear in was?

Anfrage Optimierung

- Was ist die beste Bearbeitungsreihenfolge?



  - Bearbeite in aufsteigender Häufigkeit (AND-Verknüpf)
  - Ausführung: (Caesar and Brutus) and Calpurnia
  - Schätze Größe eines OR als Summe der Einzelhäufigkeiten (Konservativ)

Beispiel

- Empfehle eine Reihenfolge

(tangerine OR trees) AND
(marmalade OR skies) AND
(kaleidoscope OR eyes)

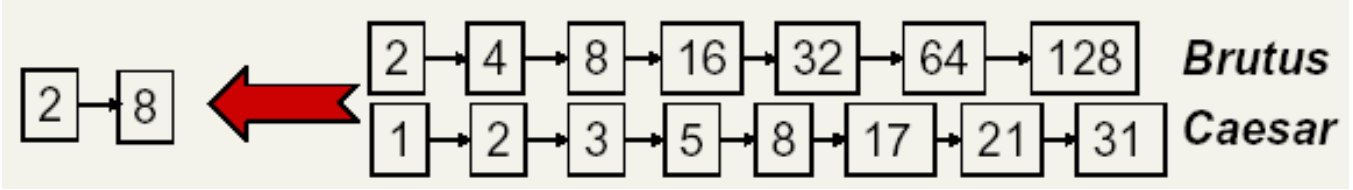
Term	Freq
eyes	213312
kaleidoscope	87009
marmalade	107913
skies	271658
tangerine	46653
trees	316812

Erweiterungen

- Skip-Pointer
- Komprimierung der Postings
- Speicherung des Dictionarys
- Kontext-Anfragen

Wiederholung, Invertierte Liste

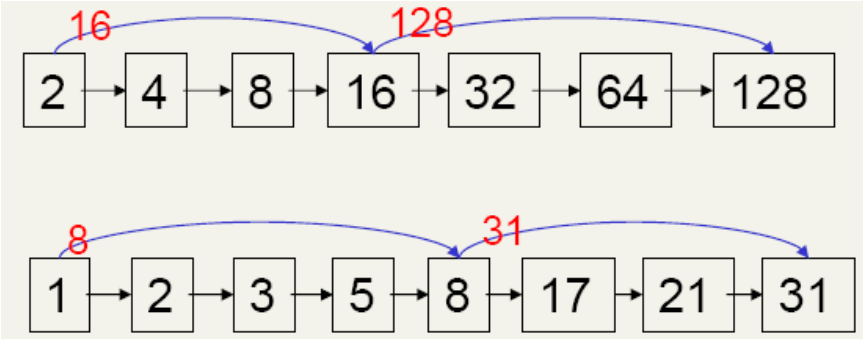
- Anfrage mit zwei Termen (logisches UND)
  - Merge-Operation durchläuft beide Listen simultan, lineare Laufzeit in Größe der Listen



  - Wenn die Listen die Größen x und y haben, die Laufzeit O(x+y)
- Können wir das verbessern?
  - Ja, wenn der Index sich nicht oft ändert

Schnelleres Zusammenfassen

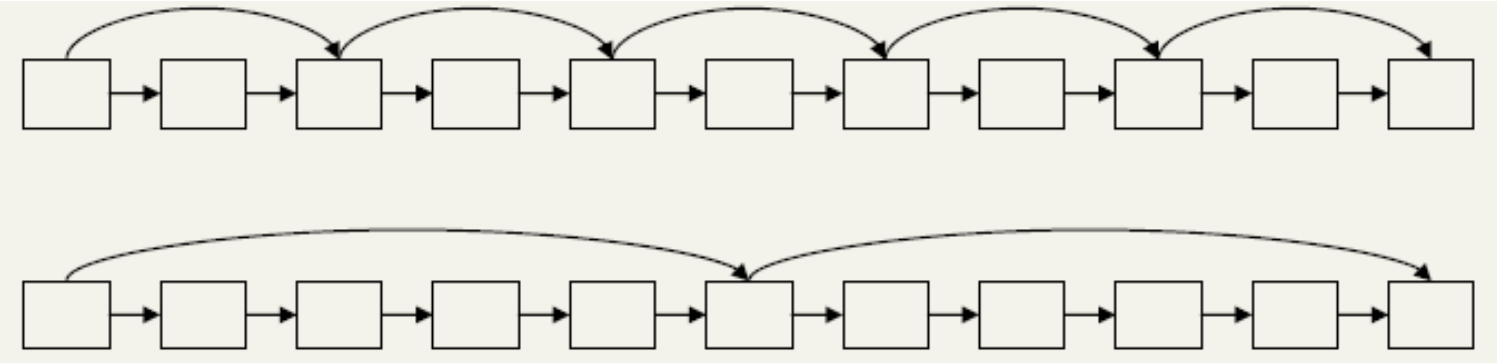
- Skip Pointer
  - überspringe Einträge, die nicht in das Ergebnis kommen



  - Warum ist das sinnvoll?
  - Um Dokument IDs zu überspringen, die nicht im Ergebnis auftauchen werden.
  - Wie?
  - Wo kommen die Skip Pointer am besten hin?

Wohin Skip Pointer plazieren? (1/2)

- Kompromiss
  - Mehr Skips -> kürzere Sprünge => wahrscheinlicher zu springen. Aber viele Zusatzvergleiche.
  - Weniger Skips => weniger Zusatzvergleiche, aber weitere Sprünge => weniger erfolgreiche Skips



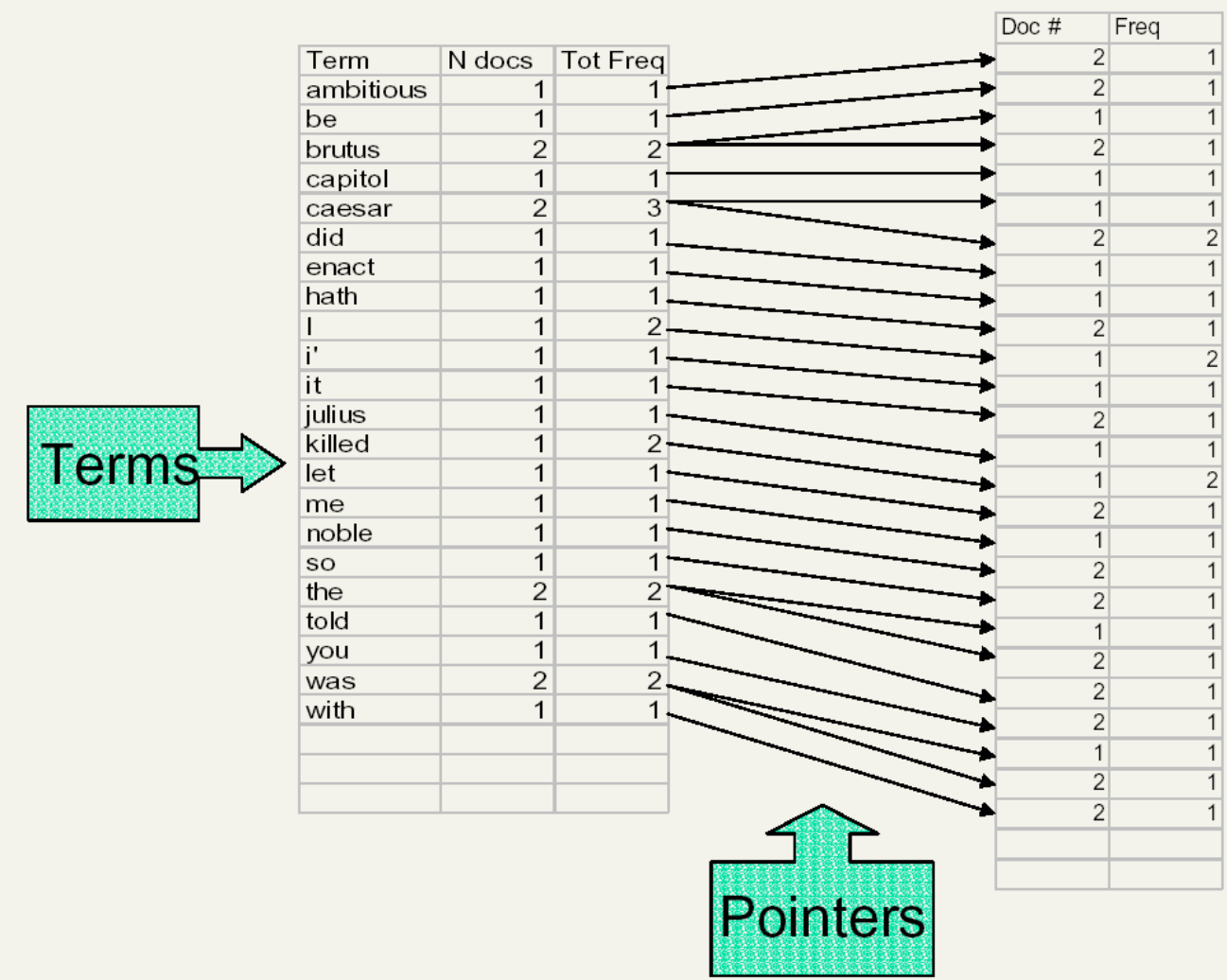
Wohin Skip Pointer plazieren? (2/2)

- Einfache Heuristik
  - Für Postings der Länge *L*,
    plaziere $\sqrt{L}$ gleichverteilte Skip Pointer
  - Ignoriert die Verteilung der Anfrageterme
  - Einfach zu implementieren, falls es wenig Änderungen im Index gibt
- Wie könnte Verteilung für Anfrageterme genutzt werden?

Rückblick

- Angenommen, wir haben
  - n = 1M Dok., jedes mit etwa 1K Termen.
  - Avg. 6 bytes/term
  - Dies macht 6GB Daten
- Angenommen, es gibt, m = 500K unterschiedliche Terme
- Term-Dokument Matrix kann nicht voll repräsentiert werden
  - $0.5*10^{12}$ 0/1-Einträge, aber nur 10^9 Einsen
  - Deshalb werden invertierte Listen genutzt
- Wo investieren wir in Speicherplatz?

Wo investieren wir in Speicherplatz?



Term	N docs	Tot Freq
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	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

Zeiger: zwei widersprüchliche Ziele

- Term Calpurnia kommt möglicherweise nur einem Dokument aus 1 Million vor
  - Platz für Zeiger $\log_2 1M \sim 20$ Bit
- Term ‘the’ kommt in nahezu jedem Dokument vor,
  - 20 Bit sind sehr teuer,
  - 0/1 Vektor wäre in dem Fall besser

Komprimierung der Postings

- Dokumentlisten werden sortiert gespeichert

  Brutus: 33,47,154,159,202 …

- Deshalb: ausreichend die Lückengröße zu speichern

  33,14,107,5,43 …

- Hoffnung: Lücken brauchen weniger als 20 Bit

Variable Kodierung der Lücken

- Ziel
  - Für Calpurnia ~20 Bits/Lücke ausgeben
  - Für „the“ ~1 Bit/Lücke ausgeben
  - Wenn die durchschnittliche Lückegröße G ist, dann soll ~$\log_2 G$ Bits/Lücke ausgeben werden
- Aufgabe
  - Kodiere jeden Integer (Lücke) im Durchschnitt mit so wenig Bits wie möglich

Gamma Kodes

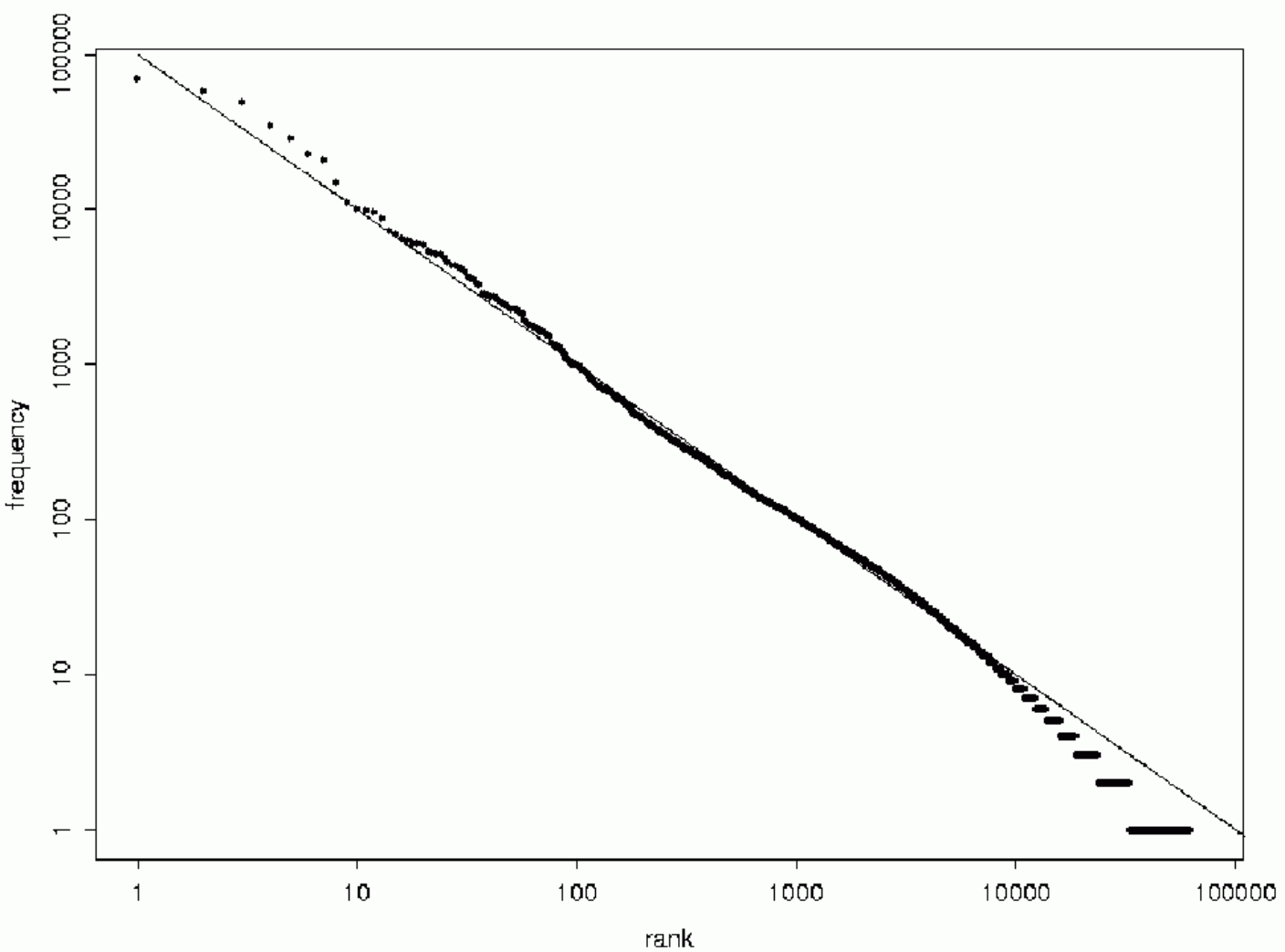
Länge | Offset

- repräsentiere Lücke G als Paar <L,O>
- Länge L ist unär und braucht $\lfloor \log_2 G \rfloor + 1$ Bits
- O $= G - 2^{\lfloor \log_2 G \rfloor}$ ist binär
- Kode für G braucht $2 \lfloor \log_2 G \rfloor + 1$ Bits
- z.B. 9 wird repräsentiert als <0001,001>
- Gamma Kode
  - Platz für Lücken: Minimum*2
- Verteilung der Lücken: Zipf Verteilung angenommen

Gamma Kode

- $1 = 2^0 + 0 = 1$
- $2 = 2^1 + 0 = 010$
- $3 = 2^1 + 1 = 011$
- $4 = 2^2 + 0 = 00100$
- $5 = 2^2 + 1 = 00101$
- $6 = 2^2 + 2 = 00110$
- $7 = 2^2 + 3 = 00111$
- $8 = 2^3 + 0 = 0001000$
- $9 = 2^3 + 1 = 0001001$
- $10 = 2^3 + 2 = 0001010$
- $11 = 2^3 + 3 = 0001011$
- $12 = 2^3 + 4 = 0001100$
- $13 = 2^3 + 5 = 0001101$
- $14 = 2^3 + 6 = 0001110$
- $15 = 2^3 + 7 = 0001111$
- $16 = 2^4 + 0 = 000010000$
- $17 = 2^4 + 1 = 000010001$

Zipf Verteilung



Platz Analyse (1/2)

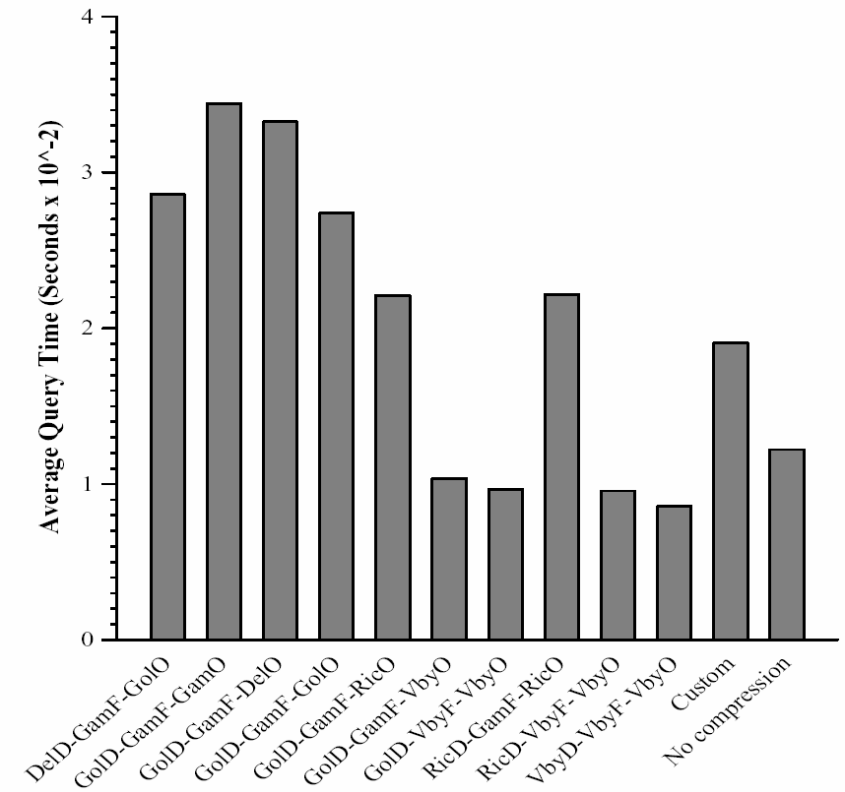
- Zipfs Gesetz
  - k-häufigste Term kommt ~1/k oft vor
- Grobe Analyse des Platz für Postings
  - häufigster Term kommt in n Dok. vor
    - n Lücken der Länge 1
  - Zweithäufigster Term kommt in n/2 Dok. vor
    - n/2 Lücken der Länge 2 ...
  - k-häufigste Term kommt in n/k Dok. vor
    - n/k Lücken der Länge k,
    - Gamma Kodes: 2logk+1Bits für jede Lücke
    - zusammen: ~(2n/k)logk Bits für den k-häufigsten Term

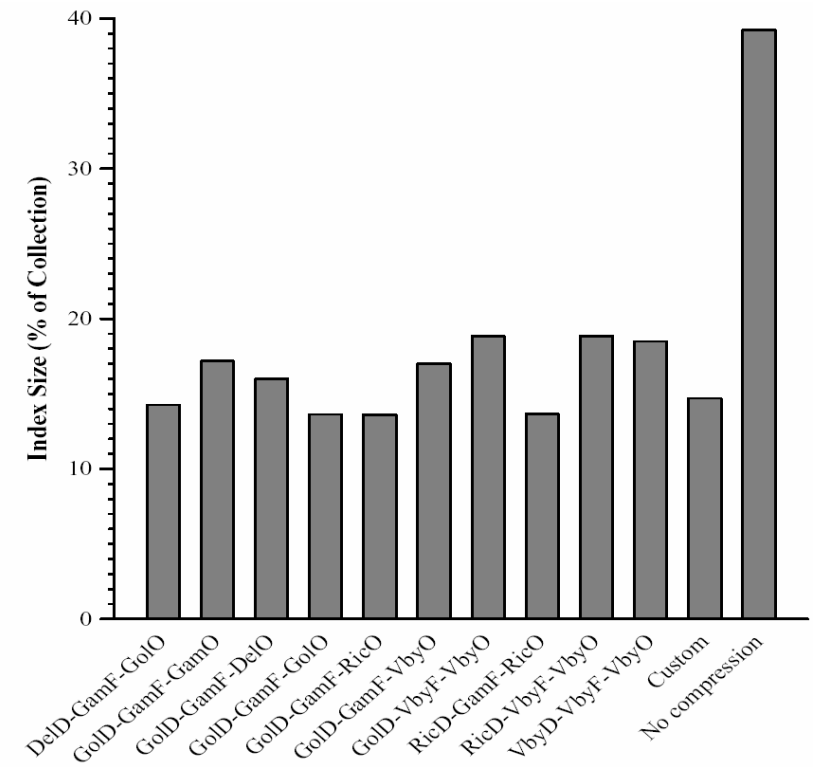
Platz Analyse (2/2)

- Summiere über k von 1 bis m=500k
- Teile die Werte für k in Gruppen
  - Gruppe i besteht aus $2^{i-1} \leq k < 2^i$
  - Gruppe i hat 2^{i-1} Summanden, jeder trägt maximal (ni)/ 2^{i-1} bei (n=1M)
- Summiert man i von 1 bis 19, bekommt man 340 MBits~ 45MB für die Postings
- Gamma Kodes sind nicht praktikabel, da Computer mit 16, 32 oder 64 Bits aufeinmal rechnen => langsam
- Word-aligned Kompression ist besser

Experimente (Scholer, Zobel, SIGIR 2002)

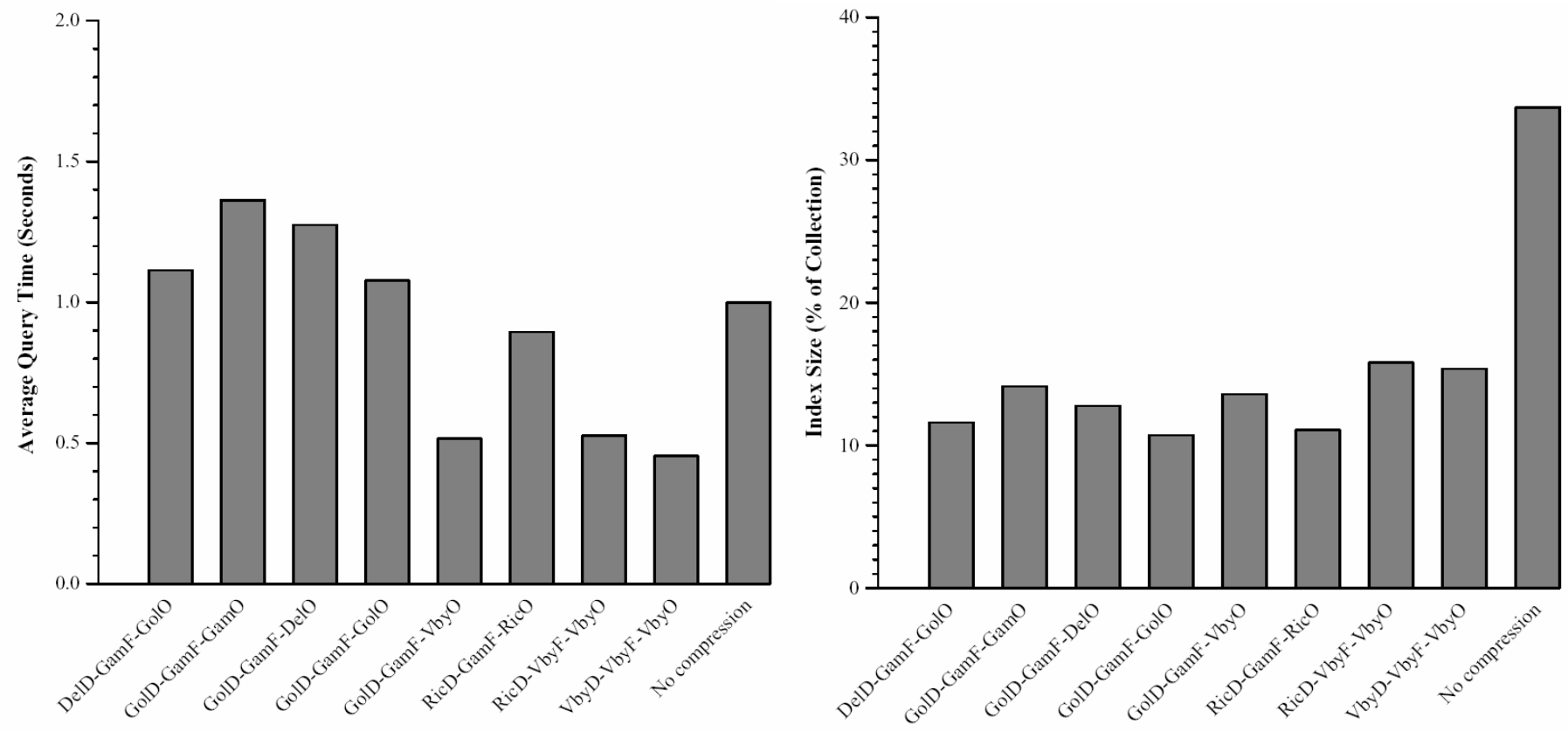
- 500 MB Daten, 10.000 Anfragen





Experimente (Scholer, Zobel, SIGIR 2002)

- 20 GB Daten, 25.000 Anfragen



Überblick

- Einführung
- Retrieval Modelle
  - Boolsches Modell
  - Vektorraummodell
  - Latent Semantic Indexing
- Evaluation
- Invertierte Listen
- Dokument Clustering, Duplikate

Dokument Clustering

- U: Raum aller möglichen Dokumente
- $D \subseteq U$: Menge von Dokumenten
- sim: $U \times U \rightarrow [0,1]$: ein Ähnlichkeitsmaß
  - Falls p,q sehr ähnlich sind, dann ist sim(p,q) nahe 1
  - Falls p,q sehr unähnlich sind, dann ist sim(p,q) nahe 0
  - Oft: sim(p,q) = 1 – d(p,q), wobei d(p,q) eine normalisierte Distanz zwischen p und q ist.
- G: ein Graph über D:
  - p,q sind durch eine Kante verbunden falls $sim(p,q) \geq t$ (t = Schwellwert)
- Ziel: finde die Zusammenhangskomponenten von G

Skizzierungsmethoden

- T = eine kleine Menge
- Eine Skizzierungsmethode für sim:
  - Kompressionsfunktion: eine zufällige Abb. ϕ: U → T
  - Rekonstruktionsfunktion: ψ: T×T → [0,1]
  - Für jedes Paar p,q gilt mit hoher Wahrscheinlichkeit

$$\psi(\phi(p),\phi(q)) \approx \text{sim}(p,q)$$

Syntaktisches Clustering mittels Skizzieren

1. P ← leere Tabelle mit Größe |D|
2. G ← leerer Graph mit |D| Knoten
3. for i = 1,…,|D|
4. Lese Dokument p_i aus der Menge
5. P[i] ← $\phi(p_i)$
6. für i = 1,…,|D|
7. für j = 1,…,|D|
8. if (ψ(P[i],P[j]) $\geq$ t)
9. füge Kante (i,j) zu G
10. Gib Zusammenhangskomponenten von G aus

Analyse

- Skizzen können in einem Durchlauf erzeugt werden
- Tabelle P kann in einer Datei auf einem Rechner gespeichert werden
- Erstellen von G erfordert $|D|^2$ Aufrufe von ψ
  - Schneller als volle Berechnung von sim
  - Quadratische Zeit ist noch ein Problem
- Zusammenhangskomponenten zu finden, ist teuer aber machbar

Locality Sensitive Hashing (LSH)

- Eine spezielle Skizzierungsmethode
- H = { h | h: U → T }: eine Familie von Hash Funktionen
- H ist locality sensitive bezüglich sim falls für alle p,q ∈ U, Pr[h(p) = h(q)] = sim(p,q).
  - Wahrscheinlichkeit ist über eine zufällige Wahl von h aus H
  - Wahrscheinlichkeit einer Kollision = Ähnlichkeit zwischen p und q

Syntaktisches Clustering mittels LSH

1. P ← leere Tabelle der Größe |D|
2. G ← leerer Graphen mit |D| Knoten
3. für i = 1,…,|D|
4. lese Dokument p_i aus der Menge
5. P[i] ← $h(p_i)$
6. sortiere P und gruppiere nach Wert
7. Gebe Gruppen aus

Analyse

- Skizzen können in einem Durchlauf erzeugt werden
- Tabelle P kann in einer Datei auf einem Rechner gespeichert werden
- Sortieren und Gruppieren erfordert O(|S| log |S|) Vergleiche
- Eine Gruppe A besteht aus dem Dokumenten mit den selben Hashwert
  - Wegen LSH Eigenschaft, die Dokument sind mit hoher Wahrscheinlichkeit ähnlich

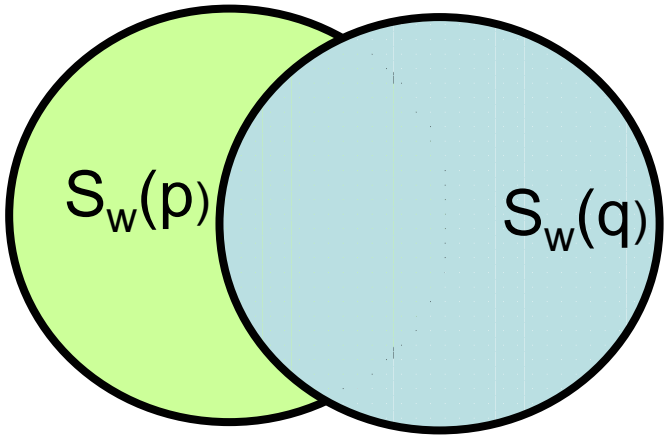
Shingling und Resemblance

- Token: Wörter, Zahlen, HTML tags, usw.
- Tokenisation(p): Sequenz der Token eines Dokuments p
- w: ein kleiner Integer
- $S_w(p)$ = w-shingling von p = Menge aller verschiedener fortlaufender Teilsequenzen von Tokenization(p) mit Länge w.
  - z.B.: p = “a rose is a rose is a rose”, w = 4
  - $S_w(p)$ = { (a rose is a), (rose is a rose), (is a rose is) }

- $\text{resemblance}_w(p,q) = \dfrac{|S_w(p) \cap S_w(q)|}{|S_w(p) \cup S_w(q)|}$

LSH für Resemblance

- $\text{resemblance}_w(p,q) = \dfrac{|S_w(p) \cap S_w(q)|}{|S_w(p) \cup S_w(q)|}$



- π = eine zufällige Permutation von Σ^w
  - π induziert eine zufällige Ordnung auf allen Tokensequenzen der Länge w
  - π induziert auch eine zufällige Ordnung auf einer beliebigen Teilmenge $X \subseteq \Sigma^w$
  - Für alle solche Teilmengen und für jedes $x \in X$, $\Pr(\min(\pi(X)) = x) = 1/|X|$
- LSH für resemblance: $h(p) = \min(\pi(S_w(p)))$

LSH für Resemblance

- Lemma: $\Pr[\min(\pi(S_w(p)) = \min(\pi(S_w(q))] = \frac{|S_w(p) \cap S_w(q)|}{|S_w(p) \cup S_w(q)|}$
- Proof:

$$
\begin{aligned}
&\Pr[\min(\pi(S_w(p)) = \min(\pi(S_w(q))] \\
&= \Pr[\min(\pi(S_w(p) \cup S_w(q)) \in S_w(p) \cap S_w(q)] \\
&= \sum_{x \in S_w(p) \cap S_w(q)} \Pr[\min(\pi(S_w(p) \cup S_w(q)) = x] \\
&= \sum_{x \in S_w(p) \cap S_w(q)} \frac{1}{|S_w(p) \cup S_w(q)|} \\
&= \frac{|S_w(p) \cap S_w(q)|}{|S_w(p) \cup S_w(q)|}
\end{aligned}
$$