

---

## Boolesche und Vektorraum-Modelle

Viele Folien in diesem Abschnitt sind eine deutsche Übersetzung der Folien von Raymond J. Mooney (<http://www.cs.utexas.edu/users/mooney/ir-course/>).

1

---

## Klassen von Retrieval-Modellen

- Boolesche Modelle (Mengen-basiert)
  - Erweitertes Boolesches Modell
- Vektorraummodelle (vector space) (statistisch-algebraischer Ansatz)
  - Latente Semantische Indexierung
- Wahrscheinlichkeitsbasierte Modelle

3

---

## Retrieval-Modelle

- Ein Retrieval-Modell spezifiziert die Details der
  - Repräsentation von Dokumenten,
  - Repräsentation von Anfragen,
  - Retrievalfunktionund legt so die Interpretation von “Relevanz” fest.
- Relevanz kann binär sein oder eine Reihenfolge angeben (*ranked retrieval*).

2

---

## Weitere Modell-Dimensionen

- Logische Sicht auf die Dokumente
  - Indexterme
  - Volltext
  - Volltext + Struktur (z.B. Hypertext)
- Anwenderaufgabe
  - Retrieval
  - Browsing

4

## Retrieval-Aufgaben

---

- **Ad-hoc-Retrieval**: Fester Dokumentenkörper, verschiedene Anfragen.
- **Filtern**: Feste Anfrage, fortlaufender Dokumentenstrom.
  - Anwenderprofil: Ein Modell mit relativ statischen Präferenzen.
  - Binäre Entscheidung: relevant/nicht relevant.
- **Routing**: Das gleiche Modell wie beim Filtern, jedoch erfolgt eine fortlaufende Bereitstellung von Ranglisten und bzgl. mehrerer Filter
  - meist in Firmen für mehrere Personen in Form von Profilen.

5

## Boolesches Modell

---

- Ein Dokument wird als eine **Menge** von Schlüsselwörtern (index terms) repräsentiert.
- Anfragen sind boolesche Ausdrücke von Schlüsselwörtern, verbunden durch AND, OR und NOT.
  - `[[[Rio & Brazil] | [Hilo & Hawaii]] & hotel & !Hilton]`
- Ausgabe: Dokument ist relevant oder nicht.  
Kein partielles Suchen, kein Ranking.

7

## Allgemeine Vorverarbeitungs-Schritte

---

- Löschen ungewollter Zeichen/Markup (z.B. HTML Tags, Interpunktion, Nummern, etc.).
- Auf Tokens anhand von Leerzeichen herunterbrechen (Schlüsselwörter)
- Wortstämme berechnen, um Wörter auf ihre Grundform abzubilden
  - computational → comput
- Stopwörter entfernen (z.B. a, the, it, etc., oder der, die, das).
- Typische Phrasen erkennen (möglicherweise durch Verwendung eines domänenspezifischen Wörterbuches).
- Invertierten Index erstellen (Schlüsselwort → Dokumentenliste, die dies enthält).

6

## Das Boolesche Retrieval-Modell

---

- Ist ein beliebtes Retrieval-Modell, da:
  - für einfache Anfragen einfach zu verstehen.
  - einfacher Formalismus.
- Boolesche Modelle können erweitert werden, um Ranking einzuschließen.
- Effiziente Implementierungen für normale Anfragen möglich.

8

## Probleme Boolescher Modelle

---

- Sehr starr: AND verlangt Anwesenheit *aller* Terme; OR die von *mindestens einem*; es gibt keine Zwischenlösungen.
- Schwierig, komplexe Anwenderanfragen auszudrücken.
- Schwierig, die Anzahl der abgerufenen Dokumente zu steuern.
  - *Alle* passenden Dokumente werden zurückgegeben.
- Schwierig, Ergebnis einzuordnen.
  - *Alle* passenden Dokumente passen logisch auf die Anfrage.  
→ keine weitere Bewertung.
- Schwierig, Relevanz-Feedback zu integrieren.
  - Falls ein Dokument vom Anwender als relevant oder irrelevant identifiziert wird, wie sollte die Anfrage geändert werden?

9

## Statistisches Retrieval

---

- Retrieval basierend auf *Ähnlichkeit* zwischen Anfrage und Dokumenten.
- Output: Dokumente, nach ihrer Ähnlichkeit zur Anfrage geordnet
- Ähnlichkeit basierend auf Auftretens-*Häufigkeiten* von Schlüsselwörtern in der Anfrage und im Dokument.
- Automatisches Relevanz-Feedback kann unterstützt werden durch:
  - Relevante Dokumente werden zur Anfrage “addiert”.
  - Irrelevante Dokumente werden von der Anfrage “subtrahiert”.

11

## Statistische Modelle

---

- Ein Dokument wird typischerweise als *bag of words (Sack von Wörtern)* (ungeordnete Liste von Wörtern und deren Häufigkeiten) repräsentiert.
- Sack = Multimenge, d.h. eine Menge, die das mehrfache Vorkommen des gleichen Elementes erlaubt.
- Anfrage = Anwender spezifiziert eine Menge gewünschter Terme (mit optionalen Gewichten):
  - Gewichtete Anfrageterme:  
 $Q = \langle \text{Datenbank } 0.5; \text{ Text } 0.8; \text{ Information } 0.2 \rangle$
  - Ungewichtete Anfrageterme:  
 $Q = \langle \text{Datenbank}; \text{ Text}; \text{ Information} \rangle$
  - Boolesche Bedingungen können in der Anfrage nicht spezifiziert werden.

10

## Probleme des Vektorraum-Modells

---

- Wie können wichtige Worte in einem Dokument bestimmt werden?
  - Wortbedeutung?
  - Wort n-Gramm (und Phrasen, Idiome,...) → Terme
- Wie kann der Grad der Wichtigkeit eines Terms innerhalb eines Dokuments und innerhalb des kompletten Korpus bestimmt werden?
- Wie kann der Grad der Ähnlichkeit zwischen einem Dokument und der Anfrage bestimmt werden?
- Im Falle des Webs: Was ist die Dokumentensammlung und was sind die Auswirkungen von Links, Formatierungsinformationen, etc.?

12

## Das Vektorraum-Modell

- Gehe davon aus, dass  $t$  eindeutige Terme nach der Vorverarbeitung bleiben; nenne sie Indexterme oder das Vokabular.
- Diese “orthogonalen” Terme spannen einen Vektorraum mit Dimension  $t$  auf.
- Jedem Term  $i$  in einem Dokument oder einer Anfrage  $j$  wird ein reellwertiges Gewicht  $w_{ij}$  zugeordnet (im einfachsten Fall die Anzahl des Auftretens von  $i$  in  $j$ ).
- Sowohl Dokumente als auch Anfragen werden als  $t$ -dimensionale Vektoren ausgedrückt:

$$d_j = (w_{1j} \ w_{2j} \ \dots \ w_{ij})$$

13

## Dokumentensammlung

- Eine Sammlung von  $n$  Dokumenten kann im Vektorraummodell durch eine Term-Dokument-Matrix dargestellt werden.
- Ein Eintrag in der Matrix entspricht dem “Gewicht” eines Terms in dem Dokument; Null heisst, dass der Term im Dokument keine Bedeutung hat oder dass er im Dokument einfach nicht vorkommt.

$$\begin{pmatrix} & T_1 & T_2 & \dots & T_t \\ D_1 & w_{11} & w_{21} & \dots & w_{t1} \\ D_2 & w_{12} & w_{22} & \dots & w_{t2} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \dots & w_{tn} \end{pmatrix}$$

15

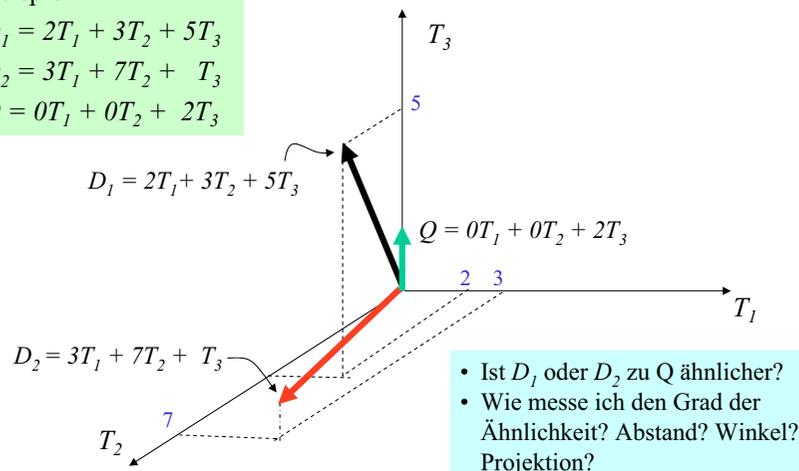
## Grafische Darstellung

Beispiel:

$$D_1 = 2T_1 + 3T_2 + 5T_3$$

$$D_2 = 3T_1 + 7T_2 + T_3$$

$$Q = 0T_1 + 0T_2 + 2T_3$$



14

## Termgewichte: Termhäufigkeit

- Häufigere Terme in einem Dokument sind wichtiger, d.h. indikativer für das Thema.

$$f_{ij} = \text{Häufigkeit von Term } i \text{ in Dokument } j$$

- Man kann Termhäufigkeit ( $tf$ ) über den gesamten Korpus normalisieren mit:

$$tf_{ij} = f_{ij} / \max_{k,l} \{f_{kl}\}$$

16

## Termgewichte: Invertierte Dokumenthäufigkeit

- Terme, die in vielen *verschiedenen* Dokumenten auftreten, sind *weniger* indikativ für das Gesamtthema.

$df_i$  = Dokumenthäufigkeit des Terms  $i$

= Anzahl der Dokumente, die Term  $i$  enthalten  
(document frequency,  $df$ )

$idf_i$  = Invertierte Dokumenthäufigkeit von Term  $i$ ,

=  $\log_2(N/df_i)$

( $N$ : gesamte Anzahl von Dokumenten)

(inverted document frequency,  $idf$ )

- Eine Angabe zur *Unterscheidungsfähigkeit* eines Terms.
- Der Logarithmus wird benutzt, um die Auswirkung bezüglich  $tf$  zu dämpfen.

17

## TF-IDF Gewichtung

- Ein typischer zusammenhängender Indikator für die Wichtigkeit eines Terms ist *tf-idf Gewichtung*:

$$w_{ij} = tf_{ij} \cdot idf_i = tf_{ij} \cdot \ln(N/df_i)$$

- Einem Term, der häufig im Dokument, aber selten im Rest der Sammlung auftritt, wird hohes Gewicht gegeben.
- Viele andere Wege zur Bestimmung von Termgewichten wurden vorgeschlagen.
- Experimentell konnte gezeigt werden, dass *tf-idf* gut funktioniert.

18

## Berechnung TF-IDF -- Ein Beispiel

- Gegeben sei ein Dokument, das Terme mit den folgenden Häufigkeiten enthält:  
A(3), B(2), C(1)
- Die Sammlung enthält 10,000 Dokumente und die Dokumenthäufigkeiten dieser Terme seien:  
A(50), B(1300), C(250)

Dann ist:

A:  $tf = 3/3$ ;  $idf = \ln(10000/50) = 5.3$ ;  $tf-idf = 5.3$

B:  $tf = 2/3$ ;  $idf = \ln(10000/1300) = 2.0$ ;  $tf-idf = 1.3$

C:  $tf = 1/3$ ;  $idf = \ln(10000/250) = 3.7$ ;  $tf-idf = 1.2$

19

## HTML-Struktur & Merkmalgewichtung

- Gewichte ggf. Tokens unter bestimmten HTML-Tags stärker:
  - `<TITLE>` Token (Google scheint Titelübereinstimmungen zu mögen)
  - `<H1>`, `<H2>`... Token
  - `<META>` Schlüsselwort-Token
- Zerlege eine Seite in verschiedene Abschnitte (z.B. Navigationsleiste und Seiteninhalt) und gewichte auf den unterschiedlichen Abschnitten basierende Token unterschiedlich.

20

## Anfragevektor

- Der Anfragevektor wird typischerweise als Dokument behandelt und ebenfalls mit tf-idf gewichtet.
- Eine Alternative für den Anwender ist, die Gewichte für die Anfrage direkt anzugeben.

21

## Ähnlichkeitsmaß

- Ein **Ähnlichkeitsmaß** ist eine Funktion, die den *Grad der Ähnlichkeit* zwischen zwei Vektoren berechnet.
- Benutzung eines Ähnlichkeitsmaßes zwischen der Anfrage und jedem Dokument:
  - Es ist möglich, die gewonnenen Dokumente in der Reihenfolge der vermuteten Bedeutung zu klassifizieren.
  - Es ist möglich, eine bestimmte Schwelle anzugeben, so dass die Größe der erhaltenen Menge an Dokumenten gesteuert werden kann.

22

## Ähnlichkeitsmaß: Skalarprodukt

- Ähnlichkeit zwischen Vektoren für das Dokument  $d_j$  und Anfrage  $q$  können berechnet werden als das Skalarprodukt (inner product) der beiden Vektoren:

$$\text{sim}(d_j, q) = d_j \cdot q = \sum_{i=1}^t w_{ij} w_{iq}$$

wobei  $w_{ij}$  das Gewicht von Term  $i$  in Dokument  $j$  und  $w_{iq}$  das Gewicht von Term  $i$  in der Anfrage ist

- Für binäre Vektoren ist das Skalarprodukt die Anzahl der übereinstimmenden Anfrageterme in einem Dokument (Größe der Schnittmenge).
- Für gewichtete Termvektoren ist es die Summe der Produkte der Gewichte der passenden Terme.

23

## Eigenschaften des Skalarproduktes

- Das Skalarprodukt hat keine obere Schranke.
- Es favorisiert lange Dokumente mit einer großen Anzahl von unterschiedlichen Termen.
- Es misst, wieviel Terme übereinstimmen, aber nicht, wie viele Terme *nicht* passen.

24

## Skalarprodukt – Beispiele

Binär:

$D = 1, 1, 1, 0, 1, 1, 0$   
 $Q = 1, 0, 1, 0, 0, 1, 1$

Vektorgröße = Größe d. Vokabulars = 7

$\text{sim}(D, Q) = 3$

0 bedeutet, dass ein entsprechender Term nicht im Dokument oder der Anfrage enthalten ist.

Gewichtet:

$D_1 = 2T_1 + 3T_2 + 5T_3$      $D_2 = 3T_1 + 7T_2 + 1T_3$   
 $Q = 0T_1 + 0T_2 + 2T_3$

$\text{sim}(D_1, Q) = 2*0 + 3*0 + 5*2 = 10$

$\text{sim}(D_2, Q) = 3*0 + 7*0 + 1*2 = 2$

retrieval  
database  
architecture  
computer  
text  
management  
information

25

## Naive Implementierung

- Berechne für jedes Dokument der Sammlung  $D$  einen *tf-idf*-gewichteten Vektor  $d_j$  mit Länge  $|V|$  ( $V$ =Vokabular).
- Konvertiere die Anfrage in einen *tf-idf*-gewichteten Vektor  $q$ .
- Für jedes  $d_j$  in  $D$  berechne Wert (Score)  $s_j = \text{cosSim}(d_j, q)$
- Sortiere Dokumente nach abnehmendem Score.
- Präsentiere dem Anwender die besten Dokumente.

Zeitkomplexität:  $O(|V| \cdot |D|)$  **Schlecht für größere  $V$  und  $D$  !**

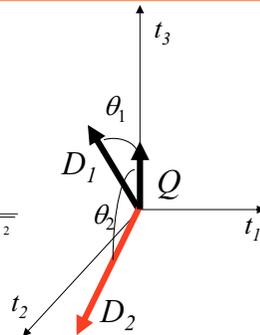
$|V| = 10,000; |D| = 100,000; |V| \cdot |D| = 1,000,000,000$

27

## Das Kosinus-Ähnlichkeitsmaß

- Kosinus-Ähnlichkeit misst den Kosinus des Winkels zwischen zwei Vektoren.
- Dies ist das Skalarprodukt normalisiert durch die Vektorlänge.

$$\text{CosSim}(d_j, q) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \cdot |\vec{q}|} = \frac{\sum_{i=1}^l (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^l w_{ij}^2} \cdot \sqrt{\sum_{i=1}^l w_{iq}^2}}$$



$D_1 = 2T_1 + 3T_2 + 5T_3$      $\text{CosSim}(D_1, Q) = 10 / \sqrt{(4+9+25)(0+0+4)} = 0.81$   
 $D_2 = 3T_1 + 7T_2 + 1T_3$      $\text{CosSim}(D_2, Q) = 2 / \sqrt{(9+49+1)(0+0+4)} = 0.13$   
 $Q = 0T_1 + 0T_2 + 2T_3$

$D_1$  ist 6 mal besser als  $D_2$  wenn die Kosinusähnlichkeit verwendet wird, aber nur 5 mal besser bei Verwendung des Skalarprodukts.

26

## Kommentare zum Vektorraum-Modell

- Einfacher, mathematisch basierter Ansatz.
- Berücksichtigt sowohl lokale (*tf*) als auch globale (*idf*) Wortauftretenshäufigkeiten.
- Liefert Ergebnisse, die nicht unbedingt alle Suchbegriffe enthalten.
- Liefert eine geordnete Liste.
- Funktioniert – trotz offensichtlicher Schwächen – in der Praxis ziemlich gut.
- Ermöglicht eine effektive Implementierung für große Dokumentensammlungen.

28

# Probleme mit Vektorraum-Modellen

---

- Fehlende semantische Informationen (z.B. Wortbedeutung, Negation).
- Fehlende syntaktische Informationen (z.B. Phrasenstruktur, Wortreihenfolge, Bereichsinformationen).
- Setzt Termunabhängigkeit voraus (ignoriert z.B. Synonyme).
- Weniger Kontrolle als im booleschen Modell (z.B. die Forderung, dass ein Term im Dokument erscheinen *mus*).
  - Bei gegebener Anfrage mit zwei Termen “A B” könnte es sein, dass ein Dokument, das A häufig enthält, aber B gar nicht, einem Dokument, das sowohl A und B enthält, aber weniger häufig, vorgezogen wird.