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/).

Retrieval Modelle

- Ein Retrieval-Modell spezifiziert die Details der:
  - Repräsentation von Dokumenten
  - Repräsentation von Anfragen
  - Retrievalfunktion
- Legt die Notation des Relevanzbegriff fest
- Relevanzbegriff kann binär oder fortlaufend sein (d.h. *ranked retrieval*).

Klassen von Retrieval Modellen

- Boolesche Modelle (Mengentheorie)
  - Erweitertes Boolesches Modell
- Vektorraummodelle (vector space) (statistische/algebraische)
  - Verallgemeinertes VS
  - Latente Semantische Indexierung
- Wahrscheinlichkeitsbasierte Modelle

Weitere Modell-Dimensionen

- Logische Sicht von Dokumenten
  - Indexterme
  - Volltext
  - Volltext + Struktur (z.B. Hypertext)
- Anwenderaufgabe
  - Retrieval
  - Browsing

Retrieval Aufgaben

- Ad hoc Retrieval: Fester Dokumentenkorpus, verschiedene Anfragen.
- Filtering: Feste Anfrage, fortlaufender Dokumentenstrom.
  - Anwenderprofil: Ein Modell mit relativ statischen Präferenzen.
  - Binäre Entscheidung: relevant/nicht relevant.
- Routing: Das gleiche Modell wie beim Filtering, jedoch erfolgt eine fortlaufende Bereitstellung von Ranglisten und nicht ein binäres Filtering.

Allgemeine Vor-Verarbeitungs-Schritte

- Löschen ungewollter Zeichen/Markup (z.B. HTML Tags, Interpunktion, Nummern, etc.).
- Auf Tokens anhand von Leerzeichen herunterbrechen (Schlüsselwörter)
- Wortstämme berechnen um Worte 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).
- Invertierte Indexe erstellen (Schlüsselwort → Dokumentenliste, die dies enthält).

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, einschließlich der Verwendung von Klammern zur Anzeige des Anwendungsbereiches.
  - [[[Rio & Brazil] | [Hilo & Hawaii]] & hotel & !Hilton]
- Ausgabe: Dokument ist relevant oder nicht.
  Kein partielles Suchen oder Ranking.

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.

Probleme Boolescher Modelle

- Sehr starr: AND bedeutet alles; OR bedeutet beliebig.
- Schwierig, komplexe Anwenderanfragen auszudrücken.
- Schwierig, die Anzahl der abgerufenen Dokumente zu überwachen.
  - *Alle* passenden Dokumente werden zurückgegeben.
- Schwierig, Output einzuordnen.
  - *Alle* passenden Dokumente passen logisch auf die Anfrage.
- Schwierig, Relevanz Feedback zu integrieren.
  - Falls ein Dokument vom Anwender als relevant oder irrelevant identifiziert wird, wie sollte die Anfrage geändert werden?

Statistische Modelle

- Ein Dokument wird typischerweise als
  bag of words (Sack voller Worte)
  (ungeordnete Liste von Worten und deren Häufigkeiten) repräsentiert.
- Sack = Menge, die das mehrfache Vorkommen des gleichen Elementes erlaubt.
- Anwender spezifiziert eine Menge gewünschter Terme mit optionalen Gewichten :
  - Gewichtete Anfrageterme:
    Q = < Datenbank 0.5; Text 0.8; Information 0.2 >
  - Ungwichtete Anfrageterme:
    Q = < Datenbank; Text; Information >
  - Keine boolesche Bedingungen in der Anfrage spezifiziert.

Statistisches Retrieval

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

Probleme bei Vektorraum-Modellen

- Wie können wichtige Worte in einem Dokument bestimmt werden?
  - Wordbedeutung?
  - 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 eine Dokumentensammlung und was sind die Auswirkungen von Links, Formatierungsinformationen, etc.?

Das Vektorraum-Modell

- Gehe davon aus, das *t* eindeutige Terme nach der Vorverarbeitung bleiben; nenne sie Indexterme oder das Vokabular.
- Diese “orthogonalen” Terme bilden einen Vektorraum.

  Dimension = t = |vocabulary|
- Jeder Term *i*, in einem Dokument oder einer Anfrage *j*, wird ein reelwertiges Gewicht $w_{ij.}$ gegeben
- Sowohl Dokumente als auch Anfragen werden als *t*-dimensionale Vektoren ausgedrückt:

  $d_j = (w_{1j}, w_{2j}, \ldots, w_{tj})$

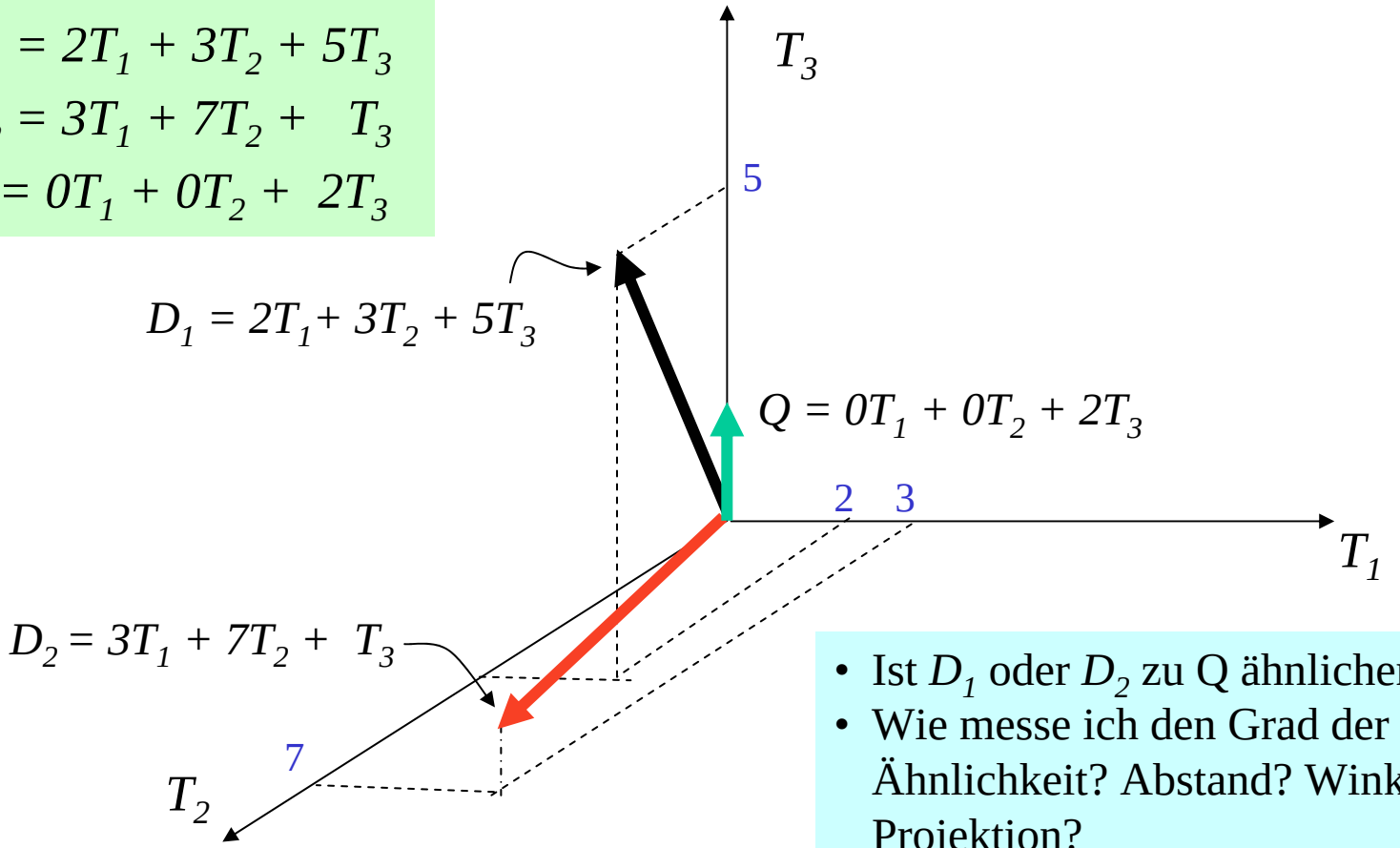
Grafische Darstellung

Beispiel:

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

$D_2 = 3T_1 + 7T_2 + T_3$

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



- Ist D_1 oder D_2 zu Q ähnlicher?
- Wie messe ich den Grad der Ähnlichkeit? Abstand? Winkel? Projektion?

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.

$$\begin{array}{c|cccc} & T_1 & T_2 & \ldots & T_t \\ D_1 & w_{11} & w_{21} & \ldots & w_{t1} \\ D_2 & w_{12} & w_{22} & \ldots & w_{t2} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \ldots & w_{tn} \end{array}$$

Termgewichte: Termhäufigkeit

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

  f_{ij} = Häufigkeit von Term *i* in Dokument *j*

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

  $tf_{ij} = f_{ij} / max\{f_{ij}\}$

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 Logorithmus wird benutzt, um die Auswirkung bezüglich *tf* zu dämpfen.

TF-IDF Gewichtung

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

$$w_{ij} = tf_{ij}\, idf_i = tf_{ij} \log_2 (N/ df_i)$$

- 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.

Berechnung TF-IDF -- Ein Beispiel

- Gegeben sei ein Dokument, dass 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 = log(10000/50) = 5.3; tf-idf = 5.3

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

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

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.

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 Reihefolge der vermuteten Bedeutung zu klassifizieren.
  - Es ist möglich eine bestimmte Schwelle anzugeben, so dass die Größe der erhaltenen Menge an Dokumenten gesteuert werden kann.

Ähnlichkeitsmaß - Skalarprodukt

- Ähnlichkeit zwischen Vektoren für das Dokument $\boldsymbol{d}_i$ und Anfrage $\boldsymbol{q}$ können berechnet werden als das Skalarprodukt (inner product) der beiden Vektoren:

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

  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 Intersektion).
- Für gewichtete Termvektoren, ist es die Summe der Produkte der Gewichte der passenden Terme.

Eigenschaften des Skalarproduktes

- Das Skalarprodukt ist unbegrenzt.
- Favorisiert lange Dokumente mit einer großen Anzahl von unterschiedlichen Termen.
- Misst wieviel Terme übereinstimmen, aber nicht wie viele Terme *nicht* passen.

Skalarprodukt -- Beispiele

Binär:

	retrieval	database	architecture	computer	text	management	information
– D =	1,	1,	1,	0,	1,	1,	0
– Q =	1,	0 ,	1,	0,	0,	1,	1

sim(D, Q) = 3

Vektorgröße = Größe d. Vokabulars = 7
0 bedeutet, dass ein entsprechender Term nicht im Dokument oder der Anfrage enthalten ist

Gewichtet:

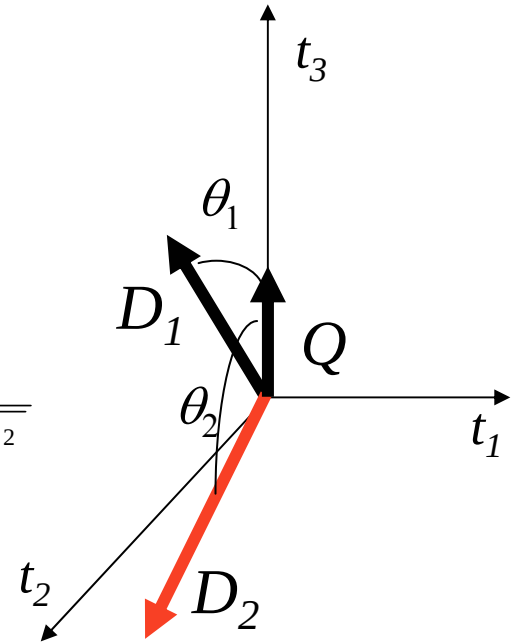
$D_1 = 2T_1 + 3T_2 + 5T_3 \qquad D_2 = 3T_1 + 7T_2 + 1T_3$
$Q = 0T_1 + 0T_2 + 2T_3$

$\text{sim}(D_1, Q) = 2*0 + 3*0 + 5*2 = 10$
$\text{sim}(D_2, Q) = 3*0 + 7*0 + 1*2 = 2$

Das Kosinus-Ähnlichkeitsmaß

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

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



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

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

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

D_1 is 6 mal besser als D_2 wobei die Kosinusähnlichkeit verwendet wird, aber nur 5 mal besser bei Verwendung des Skalarprodukt.

Naïve Implementierung

- Konvertiere alle Dokumente der Sammlung *D* in *tf-idf* gewichtete Vektoren $\boldsymbol{d}_j$, für jedes Schlüsselwort in Vokabular V.
- Konvertiere die Anfrage in einen *tf-idf*-gewichteten Vektor $\boldsymbol{q}$.
- Für jedes $\boldsymbol{d}_j$ in *D*

  berechne Wert (Score) $s_j = \text{cosSim}(\boldsymbol{d}_j, \boldsymbol{q})$
- Sortiere Dokumente nach abnehmendem Score.
- Präsentiere dem Anwender die besten Dokumente.

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

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

Kommentare zum Vektorraum-Modell

- Einfacher, mathematisch basierter Ansatz.
- Berücksichtigt sowohl lokale (*tf*) als auch globale (*idf*) Wortauftretenshäufigkeiten.
- Liefert Ergebnisse mit teilweise übereinstimmende Strings und einer geordneten Liste.
- Neigt trotz offensichtlicher Schwächen dazu, in der Praxis ziemlich gut zu arbeiten.
- Ermöglicht eine wirkungsvolle Implementierung für große Dokumentensammlungen.

Probleme mit Vektorraum-Modellen

- Fehlende semantische Informationen (z.B. Wortbedeutung).
- Fehlende syntaktische Informationen (z.B. Phrasenstruktur, Wortreihenfolge, Bereichsinformationen).
- Voraussetzung von Termunabhängigkeit (z.B. ignoriert Synonyme).
- Fehlen der Kontrolle eines booleschen Modells (z.B., *die Forderung danach*, dass ein Term im Dokument erscheinen muss).
  - Bei gegebener Anfrage mit zwei Termen “A B”, könnte es sein, dass ein Dokument, dass A häufig enthält, aber nicht B, einem Dokument, dass sowohl A und B enthält, aber weniger häufig, vorgezogen wird.