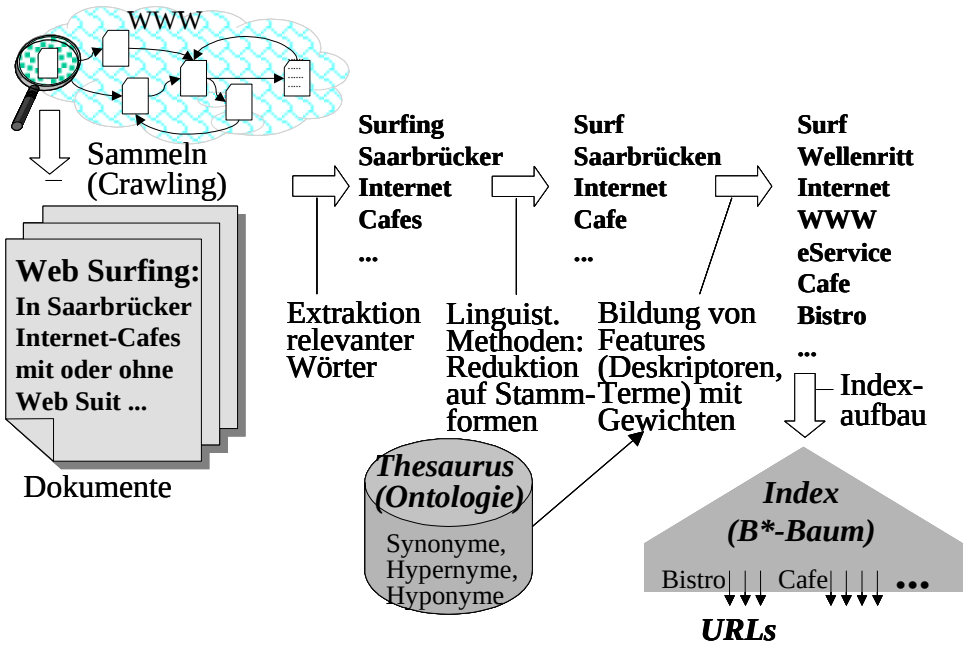
Kapitel 15: Suchmaschinen für Intranets und das Web

15.1 Grundlagen des Information Retrieval

Information-Retrieval-Systeme (IRS) verwalten große Sammlungen unstrukturierter Daten oder semistrukturierter Daten, insbesondere Textdokumente und HTML-Dokumente, und unterstützen vor allem die Suche nach relevanten Daten zu einem spezifizierten Thema bzw. ähnlichen Dokumenten zu einer spezifizierten Anfrage (einem "Musterdokument"). IRS-Funktionalität findet sich in Suchmaschinen für das Web und für Intranets, in digitalen Bibliothekssystemen, Mail-Servern und z.T. auch integriert oder als Erweiterungskomponente in objekt-relationalen Datenbanksystemen (z.B. Oracle8i interMedia). Das Prinzip der Ähnlichkeitssuche ist auch für Multimedia-Anwendungen von großer Bedeutung, z.B. für elektronische Museen u.ä.

Die folgende Abbildung gibt einen Überblick über den Aufbau und die Arbeitsweise eines IRS:



Ein *Crawler* traversiert Dokumente, im Web und in Intranets durch Verfolgen von Hyperlinks mittels HTTP-Aufrufen, und lädt sie zur Inhaltserschließung auf den IRS-Server. Eine oberflächliche *Textanalyse* extrahiert aus einem Dokument alle relevanten Wörter, ggf. durch sog. *Stoppworteliminati-on*, das Entfernen von "Füllwörtern" wie Artikel, Präpositionen, Konjunktionen. Mit einfachen linguistischen Verfahren, sog. *Stemming*-Verfahren, werden Wörter auf ihre jeweiligen Stammformen reduziert (z.B. Plural auf Singular, Dativ auf Nominativ, Partizip Perfekt auf Infinitiv eines Verbs). Die so entstandene Menge von für das Dokument signifikanten Wortstammformen wird auch als Menge von *Termen*, *Deskriptoren* oder *Features* bezeichnet. Ggf. kann diese Menge durch Nachschlagen in einem Wörterbuch, einem sog. *Thesaurus* bzw. einer sog. *Ontologie*, um Synonyme oder eng verwandte Unterbegriffe (Hyponyme) und Oberbegriffe (Hypernyme) erweitert werden. Die in einem Dokument vorkommenden Terme werden typischerweise *gewichtet*, und zwar aufgrund ihrer Vorkommenshäufigkeit und/oder aufgrund ihrer Position und Präsentation im Dokument (z.B. in

einer Überschrift, in einem großen Font, usw.); Techniken zur Gewichtung werden in Abschnitt 15.2 genauer betrachtet. Schließlich wird ein *Index* über diesen Termen als Suchschlüssel aufgebaut, typischerweise in Form eines B*-Baums (oder eines sonstigen Suchbaums für Sekundärspeicher), der mit jedem Term eine Menge von Dokument-Ids bzw. URLs sowie die Gewichte des Terms in den jeweiligen Dokumenten verbindet.

Anfragen an ein IRS sind - in der am meisten verbreiteten Form - zunächst einfache Mengen oder Listen von Termen, z.B. "Java Lava"; bei einer Interpretation als Liste werden die Terme der Anfrage unterschiedlich stark gewichtet. Das Resultat ist entweder eine Menge von Trefferdokumenten, die alle angegebenen Terme enthalten, oder eine Rangliste von zur Anfrage ähnlichen Dokumenten, die nach einem Ähnlichkeitsmaß bzw. einem Relevanz-Score (auch RSV = Retrieval Status Score genannt) absteigend sortiert ist. Ersteres nennt man *Boolesches Retrieval*, letzteres *Ranked Retrieval*. Beim Ranked Retrieval qualifizieren sich oft auch Dokumente, die nur einen Teil der Terme der Anfrage enthalten. Darüber hinaus werden u.U. auch Dokumente berücksichtigt, die keinen der Suchterme wörtlich enthalten, aber Synonyme dazu oder semantisch eng verwandte Begriffe enthalten; für diese Art des Semantischen Retrievals benötigt man einen Thesaurus bzw. eine Ontologie.

Erweiterte Formen von Anfragen erlauben die Angabe negativer Terme, die als Ausschlusskriterium interpretiert werden, z.B. "Java Lava -Coffee", wenn man keine Dokumente über die "heißesten Kaffeesorten" bekommen möchte. Darüber hinaus sind Boolesche Kombinationen von Suchbedingungen möglich und zusätzliche Bedingungen, die auf die Nachbarschaft bzw. textuelle Distanz von Wörtern abzielen, z.B. "Java NEAR Lava". Suchbedingungen dieser Art sind z.T. auch in SQL-Systemen integriert (siehe Kapitel 5).

Bei einer Anfrage, die n verschiedene Terme enthält, extrahiert das IRS zunächst die zu diesen Termen gehörigen URL- bzw. Dokument-Id-Listen aus seinem Index. Diese bilden eine Kandidatenmenge. Anschließend werden für alle Kandidaten auf der Basis ihrer Termgewichte Ähnlichkeitsmaße zur Anfrage berechnet; dabei wird die Struktur der Termkombinationen in der Anfrage wie z.B. negative Terme oder diskunktive vs. konjunktive Verknüpfung berücksichtigt. Auf diese Weise werden Kandidaten entfernt, und es werden Relevanz-Scores und das Ranking der verbleibenden Trefferdokumente berechnet.

Die Güte eines IRS, also seines Fähigkeit, zu einer Anfrage möglichst relevante und nur relevante Dokumente zurückzuliefern, wird vor allem mit den folgenden beiden Metriken bewertet:

Fähigkeit, zu einer Anfrage *nur* relevante Dokumente zu liefern:

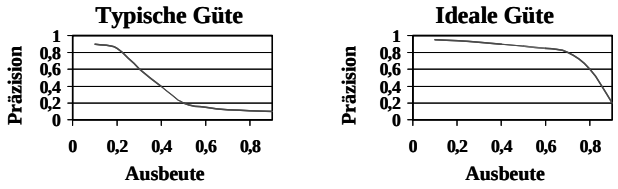
$$\textit{Präzision} \text{ einer Anfrage (engl.: } \textit{precision}) = \frac{\textit{Anzahl relevanter Dokumente unter Top } r}{r}$$

Fähigkeit, zu einer Anfrage *alle* relevanten Dokumente zu liefern:

$$\textit{Ausbeute} \text{ einer Anfrage (engl.: } \textit{recall}) = \frac{\textit{Anzahl relevanter Dokumente unter Top } r}{\textit{Anzahl aller relevanten Dokumente im Korpus}}$$

Dabei wird r typischerweise auf 10 oder 20 gesetzt, also denjenigen Präfix der Resultatsrangliste, den sich ein Benutzer in der Regel anschaut. Die Auswertung der beiden Gütemaße erfordert eine intellektuelle Bewertung der Anfrageresultate bzw. des gesamten Korpus (aller Dokumente, die das IRS verwaltet). Dies wird für spezielle Benchmark-Dokumentkollektionen (z.B. die TREC-Benchmarks) gemacht, und verschiedene IRS werden dann anhand von Anfragemixturen auf diesen Daten getestet.

Zwischen Präzision und Ausbeute gibt es einen inhärenten Zielkonflikt. Ein ideales IRS würde für beide Maße Werte in der Nähe von 1,0 erzielen, aber in der Praxis wird eine hohe Präzision oft um den Preis einer schwächeren Ausbeute und eine hohe Ausbeute oft um den Preis einer schlechteren Präzision erkauft.



15.2 Vektorraummodell für IR mit Ranking

Im Vektorraummodell nach G. Salton werden Dokumente als Vektoren in einem Feature-Raum repräsentiert. Features, also Vektorraumdimensionen, entsprechen den Termen, die in dem Korpus vorkommen, also den Wörtern nach Stoppwortelimination, Stammformreduktion und ggf. weiteren Abstraktionsschritten (siehe Abschnitt 15.1). Die Komponenten des Vektors eines Dokuments sind entweder binär und signalisieren dann schlicht das Vorkommen eines bestimmten Terms im Dokument oder reelle Zahlen aus dem Intervall [0,1] und signalisieren dann die relative Wichtigkeit oder Häufigkeit des Terms im Dokument. Diese Termgewichte werden nach bestimmten Heuristiken berechnet (siehe unten).

Anfragen sind in diesem Modell ebenfalls Vektoren im selben Featureraum. Die in der Anfrage vorkommenden Terme werden durch eine 1-Komponente im Vektor repräsentiert, alle anderen Komponenten sind 0. Wenn durch die Reihenfolge der Terme in der Anfrage implizit Gewichte ausgedrückt sein sollen (also die wichtigsten Suchterme vorne stehen müssen), können die Komponenten des Anfragevektors auch geeignet auf das Intervall [0,1] abgebildet werden. Wenn negative Terme angegeben sind, also Wörter, die in den Trefferdokumenten (möglichst) nicht vorkommen sollen, kann man auch das Intervall [-1,1] verwenden. In der Praxis werden solche Negativkriterien aber nicht im Ranking selbst behandelt, sondern als zusätzlicher Filterschritt auf der Rangliste der Kandidaten gemäß der positiven Suchterme realisiert.

Im Vektorraummodell können nun Ähnlichkeiten zwischen Dokumenten und Anfragen (oder auch zwischen verschiedenen Dokumenten) berechnet werden. Diese *Ähnlichkeitswerte*, die oft auch *Relevanz-Scores* oder *Retrieval-Status-Values (RSVs)* genannt werden, sind die Grundlage des Ranking: das Resultat ist eine nach Ähnlichkeit zur Anfrage absteigend sortierte Liste von Dokumenten. Das (zumindest als Basis für weitere Variationen) am häufigsten verwendetste Ähnlichkeitsmaß ist das *Cosinus-Maß*:

Für eine Featuremenge F, eine Dokumentmenge D, ein Dokument $d \in D \subseteq [0,1]^{|F|}$ und eine Anfrage $q \in [0,1]^{|F|}$ ist die Ähnlichkeit von d und q: $sim(d,q) = \frac{\sum_{j=1}^{|F|} d_j \, q_j}{\sqrt{\sum_{j=1}^{|F|} d_j^2} \sqrt{\sum_{j=1}^{|F|} q_j^2}}$.

Die Termgewichte der Dokumentvektoren werden in der Regel aufgrund von sog. tf*idf-Formeln bestimmt. Dabei ist tf(t,d) die Häufigkeit von Term t in Dokument d (engl. term frequency = tf), und

idf(t) ist der Kehrwert der Häufigkeit des Terms t im gesamten Korpus (engl. inverse document frequency = idf). Die Idee dahinter ist, dass das Gewicht von t in d mit seiner Häufigkeit in d wachsen soll, aber mit der Häufigkeit von t im gesamten Korpus fallen soll. Lokal häufige Terme werden also als signifikant für den Inhalt eines Dokuments angesehen, während global häufige Terme eher inflationären Charakter und daher nur noch geringe Aussagekraft über den Inhalt eines Dokuments haben. In der Praxis werden beide Größen – tf und idf – ggf. noch normalisiert (z.B. auf Werte zwischen 0 und 1), und der idf-Wert wird in der Regel logarithmisch gedämpft, da er sonst das Produkt dominieren würde. Eine typische Variante der tf*idf-Formel für das Termgewicht w_{ij} des i-ten Terms im j-ten Dokument ist z.B.:

$$w_{ij} := \frac{tf_{ij}}{max_k\ tf_{kj}} \ log\ \frac{N}{df_i}, \text{ wobei}$$

tf_{ij} die Häufigkeit von Term i in Dokument j,

N die Anzahl der Dokumente im Korpus D und

df_i die Häufigkeit von Term i im gesamten Korpus sind.

Wenn man nicht das Cosinus-Maß als Ähnlichkeitsfunktion verwendet, sonderen andere Metriken, kann auch eine Normalisierung der w_{ij}-Werte auf die Dokumentvektorlänge 1 Sinn machen:

$$\omega_{ij} := w_{ij} \ / \sqrt{\sum_k w_{kj}^2}$$

Relevanz-Feedback

Nachdem der Benutzer die ersten r Dokumente der Rangliste einer Query inspiziert hat, kann er durch Markieren der relevanten und irrelevanten Dokumente dem IRS Feedback geben, aufgrund dessen in einer weiteren Verfeinerungs- bzw. Präzisierungs-Query explizite Gewichte in der Query gesetzt werden. Im einfachsten Fall kann dies so realisiert werden, dass der Benutzer den besten Treffer auswählt und eine Query startet, die genau die Deskriptoren dieses Dokuments mit seinen entsprechenden Gewichten enthält.

Alternativ könnten - in einer längeren Recherche-Sitzung auch benutzer- bzw. sitzungsspezifische Dokumentgewichte geführt und je nach Feedback angepasst werden, was aber wesentlich aufwendiger zu implementieren ist.

15.3 Latent Semantic Indexing (LSI)

Das einfache Vektorraum-Modell ist in (mindestens) zweierlei Hinsicht zu kritisieren:

- Da es u.U. starke Korrelationen im Vorkommen verschiedener Features in den Dokumenten gibt, ist die Dimensionalität des Feature-Vektorraums eigentlich unnötig hoch.
  Beispiel:
  Wenn "Web" und "Internet" nahezu immer zusammen vorkommen, braucht man nur eines dieser beiden Features zu indexieren.
- Dasselbe Feature kann je nach vorkommender Kombination mit anderen Features eine andere Semantik haben, so daß solche Korrelation explizit berücksichtigt sein sollten.
  Beispiel:
  Wenn "Java" und "Library" zusammen auftreten, ist die Bedeutung von "Java" vermutlich die Programmiersprache Java; das Dokument behandelt, also im weiteren Sinne das Thema Internet. Wenn "Java", "Kona Blend" und "Mokka" zusammen auftreten, handelt das Dokument mit hoher Wahrscheinlichkeit von Kaffee. Wenn schließlich "Java", "Sumatra" und "Borneo" zusammen auftreten, betrifft das Dokument vermutlich das Land Indonesien.
-

Die Idee von LSI ist, solche Korrelationen auszunutzen, um von den Featurekombinationen auf Themen (Topics) bzw. Konzepte (Concepts) zu schließen, die den Inhalt von Dokumenten charakterisieren. Die Anzahl verschiedener Themen ist typischerweise deutlich kleiner als die der Features. Wenn man das Vektorraum-Modell auf Themen anwendet statt auf Features, erzeugen die Themen einen Vektorraum mit deutlich niedrigerer Dimensionalität. In den Themen kommt also die "latente Semantik" der Dokumente klarer zum Ausdruck.

Gegeben seien:
- eine Featuremenge F mit $|F| = m$
- eine Dokumentmenge D mit $|D|=n$ und $D \subseteq [0,1]^m$
- eine Query $q \in [0,1]^m$

D kann also als $n \times m$-Matrix A mit Werten aus dem Intervall [0,1] interpretiert werden und q als $1 \times m$-(Zeilen)Vektor.

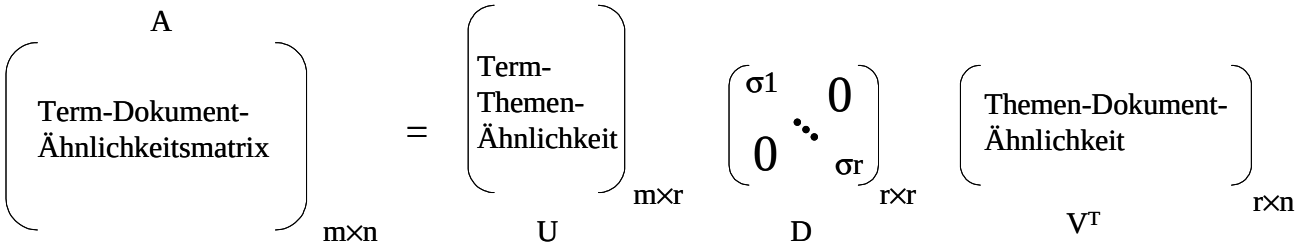
Gesucht wird:
eine "möglichst gute" Abbildung von D und q in einen Themen-Vektorraum
mit Dimensionalität $k << m$

Lösung:
Die Lösung besteht aus einer auf der sogenannten Singulärwertdekomposition von A beruhenden Transformation von A und q in einen automatisch “erzeugten" Vektorraum niedrigerer Dimensionalität. Dabei werden die folgenden mathematischen Eigenschaften der Singulärwertdekomposition ausgenutzt.

Satz:
Jede reellwertige n×m-Matrix A mit Rang r kann zerlegt werden in die Form

$$A = U \times D \times V^T$$

mit einer n×r-Matrix U mit orthonormalen Spaltenvektoren, einer r×r-Diagonalmatrix D und einer m×r-Matrix V mit orthonormalen Spaltenvektoren.
Diese Zerlegung heißt *Singulärwertdekomposition* (engl.: Singular Value Decomposition, kurz: SVD) und ist unter der Annahme, daß die Elemente von D geordnet sind, eindeutig bestimmt.



Satz:
In der Singulärwertdekomposition $A = U \times D \times V^T$ der Matrix A sind U, D und V wie folgt bestimmt:

- D besteht aus den Singulärwerten von A, d.h. den positiven Wurzeln der Eigenwerte von $A^T \times A$,
- die Spaltenvektoren von U sind die Eigenvektoren von $A \times A^T$,
- die Zeilenvektoren von V sind die Eigenvektoren von $A^T \times A$.

Dabei kann $A \times A^T$ als Dokument-Dokument-Ähnlichkeitsmatrix interpretiert werden und $A^T \times A$ als Feature-Feature-Ähnlichkeitsmatrix. U ist als *Dokument-Thema-Ähnlichkeitsmatrix* und V als *Feature-Thema-Ähnlichkeitsmatrix* zu interpretieren.

Indexierung von A und Ausführung von Queries q:
Als Index zu verwalten ist die Dokument-Thema-Ähnlichkeitsmatrix V^T sowie - quasi als Domänenwissen - die Term-Thema-Ähnlichkeitsmatrix U. Letztere kann man weitgehend statisch berechnen, sobald man eine repräsentative, große Dokumentenkollektion hat. Ein neu eingefügtes Dokument d (d.h. ein m×1-Spaltenvektor) muß dann zur Indexierung in den Themen-Vektorraum abgebildet werden, und zwar durch die Transformation $d' = U^T \times D$, und der resultierende r×1-Spaltenvektor d' wird zum Index hinzugefügt, d.h. V^T wird um eine Spalte erweitert (*Folding-in*).

Eine Query q, die als 1×m-Zeilenvektor aufgefasst werden kann, wird dann zunächst in eine Query q' in den Themen-Vektorraum transformiert, und zwar durch $q' = q \times U$.

Dann wird q' aufgrund des Index V^T evaluiert (mit der verwendeten Ähnlichkeitsfunktion des r-dimensionalen Themen-Vektorraums), und die dadurch erzeugte Rangliste von Dokumenten ist das Anfrageresultat. Im einfachsten Fall würde z.B. das Cosinus-Maß oder die Euklidische Distanz zwischen q' und den Spalten von V^T (also den Dokumenten) verwendet.

Beispiel:

m=5 (Bush, Schröder, Korea, Klose, Völler), n=7

$$A = \begin{pmatrix} 1&2&1&5&0&0&0 \\ 1&2&1&5&0&0&0 \\ 1&2&1&5&0&0&0 \\ 0&0&0&0&2&3&1 \\ 0&0&0&0&2&3&1 \end{pmatrix} = \underbrace{\begin{pmatrix} 0.58 & 0.00 \\ 0.58 & 0.00 \\ 0.58 & 0.00 \\ 0.00 & 0.71 \\ 0.00 & 0.71 \end{pmatrix}}_{U} \times \underbrace{\begin{pmatrix} 9.64 & 0.00 \\ 0.00 & 5.29 \end{pmatrix}}_{\Delta} \times \underbrace{\begin{pmatrix} 0.18 & 0.36 & 0.18 & 0.90 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.53 & 0.80 & 0.27 \end{pmatrix}}_{V^T}$$

Anfrage q = (0 0 1 0 0) wird in
q‘ = q ´ U = (0.58 0.00) transformiert und gegen V^T evaluiert.
Neues Dokument d8 = $(1\ 1\ 0\ 0\ 0)^T$ wird in
d8‘ = U^T ´ d8 = $(1.16\ \ 0.00)^T$ transformiert und an V^T angefügt.

SVD als Regressionsverfahren und für approximatives LSI

Wenn der Rang r von A nicht hinreichend klein ist, also die Anzahl der “Themen" zu groß ist, kann man die Dimensionalität des Themen-Vektorraums künstlich auf k < r beschränken, indem man nur die k größten Singulärwerte von A und die zugehörigen Eigenvektoren betrachtet. Dadurch berücksichtigt man die “ausgeprägtesten" Themen und “stärksten" Zusammenhänge zwischen Features und Themen, was durch den folgenden Satz formal untermauert wird.

Satz:
Sei A eine m×n-Matrix mit Rang r und sei $A_k = U_k \times D_k \times V_k^T$, wobei die k×k-Diagonalmatrix D_k die k größten Singulärwerte von A enthält und die m×k-Matrix U_k und die k×n-Matrix V_k^T aus den zugehörigen Eigenvektoren der Singulärwertdekomposition von A bestehen.
Unter allen m×n-Matrizen C mit einem Rang, der nicht größer als k ist, ist A_k diejenige Matrix, die den Fehler $\|A-C\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij}-C_{ij})^2$ minimiert (die Frobenius-Norm).

Zur Illustration:
Die SVD kann als affine Transformation der Dokumentvektoren aufgefaßt werden. Bei der Beschränkung auf die k größten Singulärwerte erfolgt quasi eine Projektion auf die k aussagekräftigsten Dimensionen (diejenigen mit der größten Varianz der Datenpunktkoordinaten).

Bei der Indexierung von Dokumenten und der Ausführung von Queries im - nunmehr k-dimensionalen Themenreaum - spielen dann einfach U_k und V_k die Rolle von U und V.

Beispiel:

m=6 terms t1: bak(e,ing) t2: recipe(s) t3: bread
t4: cake t5: pastr(y,ies) t6: pie

n=5 documents
d1: How to bake bread without recipes
d2: The classic art of Viennese Pastry
d3: Numerical recipes: the art of scientific computing
d4: Breads, pastries, pies and cakes: quantity baking recipes
d5: Pastry: a book of best French recipes

$$A = \begin{pmatrix} 0.5774 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\ 0.5774 & 0.0000 & 1.0000 & 0.4082 & 0.7071 \\ 0.5774 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 & 0.4082 & 0.7071 \\ 0.0000 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \end{pmatrix}$$

$$A = \begin{pmatrix} 0.2670 & -0.2567 & 0.5308 & -0.2847 \\ 0.7479 & -0.3981 & -0.5249 & 0.0816 \\ 0.2670 & -0.2567 & 0.5308 & -0.2847 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 \\ 0.5198 & 0.8423 & 0.0838 & -0.1158 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 \end{pmatrix} \quad U$$

$$\times \begin{pmatrix} 1.6950 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.1158 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.8403 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.4195 \end{pmatrix} \quad \Delta$$

$$\times \begin{pmatrix} 0.4366 & 0.3067 & 0.4412 & 0.4909 & 0.5288 \\ -0.4717 & 0.7549 & -0.3568 & -0.0346 & 0.2815 \\ 0.3688 & 0.0998 & -0.6247 & 0.5711 & -0.3712 \\ -0.6715 & -0.2760 & 0.1945 & 0.6571 & -0.0577 \end{pmatrix} \quad V^T$$

$$A_3 = \begin{pmatrix} 0.4971 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\ 0.6003 & 0.0094 & 0.9933 & 0.3858 & 0.7091 \\ 0.4971 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\ 0.1801 & 0.0740 & -0.0522 & 0.2320 & 0.0155 \\ -0.0326 & 0.9866 & 0.0094 & 0.4402 & 0.7043 \\ 0.1801 & 0.0740 & -0.0522 & 0.2320 & 0.0155 \end{pmatrix} = U_3 \times \Delta_3 \times V_3^T$$

Anfrage q: baking bread → q = (1 0 1 0 0 0)
Transformation in den Themenraum mit k=3 → q‘ = q ´ U_k = (0.5340 -0.5134 1.0616)
Skalarprodukt-Ähnlichkeit im Themenraum mit k=3:
sim (q, d1) = q‘ ´ Vk_{*1}^T » 0.8 sim (q, d2) = q‘ ´ Vk_{*2}^T » -0.2
sim (q, d3) = q‘ ´ Vk_{*3}^T » -0.2 usw.

Folding-in eines neuen Dokuments d6:
algorithmic recipes for the computation of pie → d6 = (0 0.7071 0 0 0 0.7071)^T
Transformation in den Themenraum mit k=3 → d6‘ = U_k^T ´ d6 » (0.5 -0.28 -0.15)
d6‘ als neue Spalte an Vk^T anhängen

Zusammenfassende Bewertung von LSI

LSI ist ein elegantes, mathematisch wohlfundiertes Modell, das in Benchmarks auf homogenen Korpora wie z.B. Patentsammlungen, Wirtschaftsnachrichten oder Bibliotheken sehr gute Ergebnisse bzgl. Präzision und Ausbeute liefert. Dabei wird die approximative (Regressions-) Variante mit Werten von k in der Größenordnung von 100 verwendet. Diese Variante liefert deutlich bessere Ergebnisse als die mit dem vollen Rang r, da sie gewissermaßen Rauschen unterdrückt und Korrelationen kompakter zu Themen zusammenfasst.

LSI kann sogar für multilinguales IR verwendet werden, wenn man eine hinreichend große Startmenge von Dokumenten hat, die in mehreren Sprachen verfügbar sind. Dazu werden die mehrsprachigen Varianten desselben Dokument jeweils zu einem einzigen virtuellen Dokument konkateniert, und die SVD wird durch diese Startmenge berechnet. Dadurch kommen die starken Korrelationen zwischen den Wörter desselben Begriffs in mehreren Sprachen implizit im selben Thema zum Ausdruck. Weitere Dokumente können dann auch nur in einer Teilmenge der unterstützten Sprachen (z.B. einer einzigen Sprache) verfügbar sein und werden per Folding-in in den Index eingefügt. Anfragen in einer beliebigen der unterstützten Sprachen werden auf den latenten Themenraum abgebildet und können somit auch Treffer in anderen Sprachen finden.

Für Web-Retrieval hat sich LSI nicht bewährt, da man es dort mit einem hochgradig heterogenen, terminologisch extrem streuenden Korpus zu tun hat. Hier ist es sehr schwer, einen brauchbaren Wert für die Zahl k der berücksichtigten Singulärwerte zu finden.

15.4 Linkanalyse für Autoritäts-Ranking

Zusätzlich zu ihrer inhaltlichen Relevanz können Dokumente auch nach ihrer Autorität, also nach Umfang, Klarheit und Signifikanz der in einem Dokument enthaltenen Information, bewertet werden, und entsprechende Autoritätsmaße können in das Ranking der Resultatsdokumente einer Query einfließen. Durch Kombination von Relevanz- und Autoritätsbewertung ist es möglich, die Präzision von Suchresultaten deutlich zu steigern, indem unter allen Treffern einer Anfrage diejenigen Dokumente mit hoher Autorität möglichst weit vorne in der Rangliste platziert werden.

Eine naheliegende Idee, Autorität von Web-Dokumenten zu quantifizieren, besteht darin, die eingehenden Hyperlinks einer Web-Seite als Zitate durch andere Web-Nutzer zu interpretieren. Einfach nur die Anzahl eingehender Links als Autoritätsmaß zu verwenden, greift jedoch zu kurz, da Links je nach Ursprung unterschiedliches Gewicht haben könnten und ein brauchbarer Ansatz resistent sein sollte gegen Link-Manipulationen, Zitier-Cliquen, u.ä.

15.4.1 Autoritäts-Ranking nach der Methode von Page und Brin

Das Web (oder ein Intranet) wird bei dieser Methode, die in der Suchmaschine Google verwendet wird, als gerichteter Graph G = (V,E) gesehen mit Web-Seiten als Knotenmenge V, |V|=n, und Hyperlinks als Kantenmenge E. Für die folgenden Berechnungen wird angenommen, dass der Graph komplett a priori bekannt ist und seine Adjazenzmatrix A, eine n×n-Matrix mit Aij = 1 falls $(i,j) \in E$, 0 sonst, auf einem Server gespeichert ist. Google baut diesen Graph aufgrund der Ergebnisse seines Crawlers auf; diese sind bei Google mehr als 1 Milliarde Knoten.

Die Kernidee dieser Methode ist, dass die *Autorität (Authority-Score, Authority-Rank)* r(q) einer Web-Seite q proportional zur Summe der Autoritätsbewertungen der Vorgänger von q ist – vorausgesetzt, alle Vorgänger hätten dieselbe Anzahl ausgehender Links. Bei Vorgängern mit vielen ausgehenden Kanten würde man das Autoritätsmaß nur anteilsmäßig auf die Zieldokumente der Kanten umlegen. Diese Überlegung führt auf die folgende erste Arbeitsdefinition:

$$r(q) = k \sum_{(p,q)\in E} r(p) / outdegree(p)$$

mit einer Konstanten k und der Anzahl *outdegree(p)* von p ausgehender Kanten.

Es zeigt sich jedoch, dass diese Definition noch nicht tragfähig ist, da unklar ist, wie man mehreren Zusammenhangskomponenten oder gar isolierten Knoten umgehen sollte. Daher weist die Lösung von Page und Brin jeder Web-Seite unabhängig von ihren Vorgängern eine Mindestautorität ε/n zu und berechnet die (1-ε) gewichtete Restautorität nach dem o.a. Ansatz.

Definition:
Die Autorität der Web-Seite q im Web-Graphen G=(V,E) ist gegeben durch

$$r(q) = \varepsilon / n + (1-\varepsilon) \sum_{(p,q)\in E} r(p) / outdegree(p) .$$

Dabei ist ε ein Kalibrierungsparameter, für den lt. Page und Brin $0 < \varepsilon \leq 0.2$ gelten sollte.

Satz:

Mit einer modifizierten Matrix A' mit A'_{ij} = 1/outdegree(i) falls $(i,j) \in E$ und 0 sonst, gilt $\vec{r} = \vec{\varepsilon} / n \ + (1-\varepsilon) A' \vec{r}$ und äquivalent dazu

$$\frac{1}{1-\varepsilon}\vec{r} = \vec{\varepsilon} / n \ + \left(\frac{\vec{\varepsilon}}{(1-\varepsilon)n} \vec{1}^T + A' \right) \vec{r}$$

Dabei sind $\vec{\varepsilon}$ und $\vec{1}$ n×1-Spaltenvektoren, die komplett mit dem Wert ε bzw. 1 besetzt sind.

Man sieht somit, dass der Spaltenvektor $\vec{r}$ der Autoritätswerte Eigenvektor einer modifizierten Transitionsmatrix ist. Eine mögliche, approximative und iterative, Berechnung ist:

$\vec{r}^{(0)} = \vec{1} / n$

Wiederhole bis sich die größten Werte von $\vec{r}$ kaum noch ändern:

$\vec{r}^{(i+1)} = \vec{\varepsilon} / n \ + (1-\varepsilon) A' \vec{r}^{(i)}$

Die besten Autoritäten sind dann diejenigen Komponenten von $\vec{r}$ mit den größten Werten.

Diese Methode funktioniert in der Praxis verblüffend gut. Google z.B. rechnet ca. 100 Iterationen und speichert dann die Web-Seiten-spezifischen Autoritätswerte in seinem Index. Bei Anfragen berechnet Google pro Trefferseite einen anfragespezifischen Relevanz-Score und kombiniert diesen in einer gewichteten Summe mit dem vorberechneten Autoritäts-Score. Die Resultatsliste ist der Präfix der nach dieser gewichteten Summe absteigend sortierten Liste. Die relativen Gewichte von Relevanz und Autorität sind per Trial-and-Error anhand typischer Google-Queries ermittelt.

Auch die Methode in der Praxis sehr gut funktioniert, ist vor allem der Ad-hoc-Charakter der oben eingeführten Gewichtungen, insbesondere auch der etwas willkürlichen Einführung von ε, wissenschaftlich unbefriedigend. Eine alternative Herleitung derselben Lösung verwendet einen sog. Random Walk auf dem Web-Graphen und mathematische Resultate über Markov-Ketten.

Exkurs Wahrscheinlichkeitsrechnung

Ein **Wahrscheinlichkeitsraum** ist ein Tripel (Ω, E, P) mit
- einer Menge W elementarer Ereignisse,
- einer Familie E von Teilmengen von Ω mit $\Omega \in$ E, die unter
  $\cup$, $\cap$ und - mit abzählbar vielen Operanden abgeschlossen ist
  (bei endlichem Ω ist in der Regel $E=2^{\Omega}$), und
- einem Wahrscheinlichkeitsmaß P: E $\rightarrow$ [0,1] mit P[Ω]=1 und P[$\cup_i$ Ai] = Σ_i P[Ai]
  für abzählbar viele, paarweise disjunkte Ai

Eigenschaften von P:
P[A] + P[$\neg$A] = 1
P[$\varnothing$] = 0
P[A $\cup$ B] = P[A] + P[B] – P[A $\cap$ B]
P[Ω] = 1
Eine **Zufallsvariable** X über einem Wahrscheinlichkeitsraum (W, E, P) ist eine Funktion X: $\Omega \rightarrow$ M mit M $\subseteq$ R, so daß {e | X(e) $\leq$ x} $\in$ E für alle x $\in$ M.

F_X: M $\rightarrow$ [0,1] mit $F_X(x)$ = P[X $\leq$ x] heißt Verteilungsfunktion von X;
bei abzählbarer Menge M heißt f_X: M $\rightarrow$ [0,1] mit $f_X(x)$ = P[X = x] Dichtefunktion von X,
ansonsten ist $f_X(x)$ durch $F'_X(x)$ gegeben
Zwei Ereignisse A, B eines W.raums heißen **unabhängig**, wenn gilt P[A $\cap$ B] = P[A] P[B].
Die **bedingte Wahrscheinlichkeit P[A | B]** von A unter der Bedingung (Hypothese) B ist definiert als: $P[A \mid B] = \frac{P[A \cap B]}{P[B]}$

Satz von der totalen Wahrscheinlichkeit:

Für eine Partitionierung von W in Ereignisse B1, ..., Bn gilt: $P[A] = \sum_{i=1}^{n} P[A \mid Bi]\, P[Bi]$

Ein **stochastischer Prozeß** ist eine Familie von Zufallsvariablen $\{X(t) \mid t \in T\}$.
T heißt Parameterraum, und der Definitionsbereich M der X(t) heißt Zustandsraum. T und M können diskret oder kontinuierlich sein.
Ein stochastischer Prozeß heißt **Markov-Prozeß**, wenn für beliebige $t_1, ..., t_{n+1}$ aus dem Parameterraum und für beliebige $x_1, ..., x_{n+1}$ aus dem Zustandsraum gilt:
Ein Markov-Prozeß mit diskretem Zustandsraum heißt **Markov-Kette**. O.B.d.A. werden die natürlichen Zahlen als Zustandsraum gewählt. Notation für Markov-Ketten mit diskretem Parameterraum: X_n statt $X(t_n)$ mit n = 0, 1, 2, ...
Die Markov-Kette Xn mit diskretem Parameterraum heißt
homogen, wenn die Übergangswahrscheinlichkeiten pij := $P[X_{n+1} = j \mid X_n=i]$ unabhängig von n sind
irreduzibel, wenn jeder Zustand von jedem Zustand mit positiver Wahrscheinlichkeit erreichbar ist:
$\sum_{n=1}^{\infty} P[X_n = j \mid X_0 = i] > 0$
aperiodisch, wenn alle Zustände i die Periode 1 haben, wobei die Periode von i der ggT aller Werte n ist, für die gilt: $P[X_n = i \wedge X_k \neq i \text{ für } k = 1,...,n-1 \mid X_0 = i] > 0$
Die Markov-Kette Xn mit diskretem Parameterraum heißt
positiv rekurrent, wenn für jeden Zustand i die Rückkehrwahrscheinlichkeit gleich 1 ist und mittlere Rekurrenzzeit endlich:
ergodisch, wenn sie homogen, irreduzibel, aperiodisch und positiv rekurrent ist.

Für die **n-Schritt-Transitionswahrscheinlichkeiten**Für die **Zustandswahrscheinlichkeiten nach n Schritten**
Jede homogene, irreduzible, aperiodische Markov-Kette mit endlich vielen Zuständen ist
positiv rekurrent und ergodisch
Für jede ergodische Markov-Kette existieren **stationäre Zustandswahrscheinlichkeiten**
$\pi_j := \lim_{n\to\infty} \pi_j^{(n)}$**.** Diese sind unabhängig von $P^{(0)}$und durch das folgende lineare Gleichungssystem bestimmt: $\pi_j = \sum_i \pi_i p_{ij} \quad \text{für alle } j$ (Gleichgewichtsgleichungen) und $\sum_j \pi_j = 1$

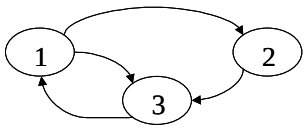
bzw. in Matrix-Notation: $\Pi^T = \Pi^T P$ und $\Pi^T \vec{1} = 1$

Autoritätsbewertung mittels Random Walk

Das Verfahren von Page und Brin modelliert einen Random Walk über den Web-Graphen, bei dem man mit Wahrscheinlichkeit (1-ε) von der aktuell besuchten Web-Seite zu einem der Nachfolger wandert und mit Wahrscheinlichkeit ε einen „Random Jump“ zu irgendeiner Web-Seite macht. Im ersten Fall wird die als nächstes besuchte Seite gemäß einer Gleichverteilung unter den Nachfolgern der aktuellen Seite ausgewählt; im zweiten Fall wird eine Seite unter allen Web-Seiten gemäß einer Gleichverteilung ausgewählt.

Für den so definierten Random Walk kann man eine Markov-Kette angeben, die beweisbar ergodisch ist. Der Autoritäts-Score einer Seite nach dem Verfahren von Page und Brin ist die **stationäre Besuchswahrscheinlichkeit der Web-Seite** in dieser Markov-Kette.

Beispiel:



$\varepsilon = 0.2$ $\quad P = \begin{pmatrix} 0.0 & 0.5 & 0.5 \\ 0.1 & 0.0 & 0.9 \\ 0.9 & 0.1 & 0.0 \end{pmatrix}$

$$\Pi^{(0)} \approx \begin{pmatrix} 0.333 \\ 0.333 \\ 0.333 \end{pmatrix} \Rightarrow \Pi^{(1)} \approx \begin{pmatrix} 0.333 \\ 0.200 \\ 0.466 \end{pmatrix} \Rightarrow \Pi^{(2)} \approx \begin{pmatrix} 0.439 \\ 0.212 \\ 0.346 \end{pmatrix} \Rightarrow \Pi^{(3)} \approx \begin{pmatrix} 0.332 \\ 0.253 \\ 0.401 \end{pmatrix}$$

$$\Rightarrow \Pi^{(4)} \approx \begin{pmatrix} 0.385 \\ 0.176 \\ 0.527 \end{pmatrix} \Rightarrow \Pi^{(5)} \approx \begin{pmatrix} 0.491 \\ 0.244 \\ 0.350 \end{pmatrix}$$

$\pi1 = 0.1\ \pi2 + 0.9\ \pi3$
$\pi2 = 0.5\ \pi1 + 0.1\ \pi3$
$\pi3 = 0.5\ \pi1 + 0.9\ \pi3$
$\pi1 + \pi2 + \pi3 = 1$ $\quad \Rightarrow \pi1 \approx 0.433,\ \pi2 \approx 0.094,\ \pi3 \approx 0.471$

15.4.2 Autoritäts-Ranking nach der HITS-Methode von Kleinberg

Bei dem HITS-Verfahren nach Kleinberg (Hyperlink-Induced Topic Search) analysiert man die Linkstruktur eines relativ kleinen Untergraphen, der aufgrund seiner thematischen Relevanz bestimmt wird. Man berechnet zunächst aufgrund einer klassischen Relevanzbewertung eine Menge von Wurzelseiten, z.B. die Top 100 einer Anfrage bei Google oder AltaVista, und fügt zu dieser alle Nachfolger und möglichst viele Vorgänger hinzu sowie alle Kanten zwischen den betrachteten Seiten; der Graph G=(V,E), |V|=n, der so enstandene Menge von – z.B. einigen Tausend – Seiten V bildet die Basis der Linkanalyse. Anders als beim Verfahren nach Page und Brin ist dieser Graph anfragespezifisch.

Die Arbeitshypothese des HITS-Verfahrens ist, dass es in einem solchen Graph außer guten *Autoritäten (Authorities)* auch gute *Referenzen (Hubs)* gibt:Linksammlungen, die Verweise auf die besten Autoritäten eines bestimmten Themas enthalten. Das Verfahren berechnet für jede Seite p im Graphen sowohl ein *Autoritätsgewicht (Authority-Score)* x_p als auch ein *Referenzgewicht (Hub-Score).* y_p. Zwischen Autoritäten und Referenzen gibt es eine wechselseitige Rekursion: eine Autorität ist umso besser, hat also höheres Referenzgewicht, je besser die Autoritäten sind, auf die sie verweist, und eine Autorität ist umso besser, hat also höheres Autoritätsgewicht, je besser die Referenzen sind, die auf sie verweisen. Wenn man postuliert, dass dieser Zusammenhang zwischen Autoritätsgewichten x_p und Referenzgewichten y_p linear ist, kommt man auf folgende Gleichungen:

$$x_q = \sum_{(p,q)\in E} y_p \quad \text{und} \quad y_p = \sum_{(p,q)\in E} x_q$$

bzw. in Matrix-Notation:

$\vec{x} = A^T \vec{y}$ und $\vec{y} = A\vec{x}$, wobei A die Adjazenzmatrix von G ist.

Setzt man die rechte Seite der y-Gleichung in die rechte Seite der x-Gleichung ein (und analog für y), so erhält man

$$\vec{x} := A^T \vec{y} := A^T A \vec{x} \quad \text{und} \quad \vec{y} := A\vec{x} := A A^T \vec{y},$$

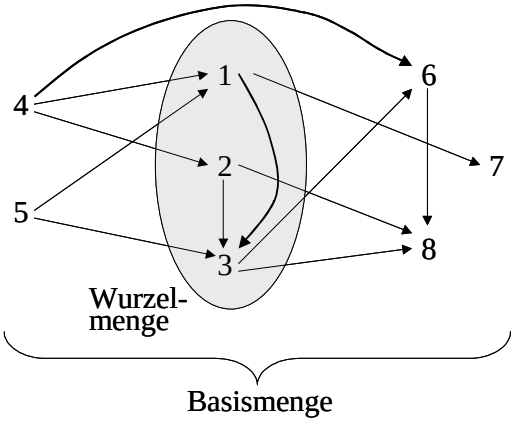
und wir erkennen, dass die Vektoren x und y Eigenvektoren der Matrizen A^TA bzw. AA^T sind.

Die Matrix $M^{(auth)} := A^TA$ kann als Cocitation-Matrix interpretiert werden: $M^{(auth)}_{ij}$ ist die Anzahl der Web-Seiten, die auf i und auf j verweisen. Die Matrix $M^{(hub)} := AA^T$ kann als Bibliographic-Coupling-Matrix interpretiert werden: $M^{(hub)}_{ij}$ ist die Anzahl der Web-Seiten, auf die sowohl i als auch j verweisen

Die wechselseitige Rekursion kann iterativ berechnet werden, indem man sowohl den x- als auch den y-Vektor zunächst mit dem Wert 1/n in allen Komponenten initialisiert und dann abwechselnd x und y neu berechnet, bis sich die größten Komponenten nur noch geringfügig ändern. In jedem Iterationsschritt werden die erhaltenen x- und y-Vektoren auf die Länge 1 normalisiert. Dieses Iterationsverfahren konvergiert (unter bestimmten Bedingungen, die in der Regel gegeben sind) gegen die Eigenvektoren von A^TA und AA^T, die zum betragsgrößten Eigenwert gehören.

Der HITS-Algorithmus sieht insgesamt also folgendermaßen aus:
1) Bestimme hinreichend viele (z.B. 50-200) „Wurzelseiten“
   per Relevanz-Ranking (z.B. mittels tf*idf-Ranking)
2) Füge alle Nachfolger von Wurzelseiten hinzu
3) Füge für jede Wurzelseite max. d Vorgänger hinzu (durch Zufallsauswahl unter allen Vorgängern)
4) Erzeuge den Graph G=(V,E), |V|=n, für die so erhaltene Basismenge
5) Initialisiere alle Komponenten von $x^{(0)}$ mit 1/n und die Komponenten von $y^{(0)}$ mit 1/n
6) Solange noch keine hinreichende Konvergenz erreicht ist
   (oder für eine feste Anzahl von Iterationen) wiederhole
   $x^{(i)} := A^T y^{(i-1)}$ und $y^{(i)} := A x^{(i-1)}$
7) Gib Seiten nach absteigend sortierten Authority-Scores aus
   (z.B. die 10 größten Komponenten von x)

Illustration der Schritte 1 bis 4:



Das HITS-Verfahren hat eine gewisse Anfälligkeit gegenüber Themendrifts: auch wenn die Wurzelmenge nur für das ursprünglich gegebene Thema (die Anfrage) relevant ist, könnte der Algorithmus auf sehr starke Autoritäten und Referenzen zu einem anderen Thema stoßen, so dass das Endresultat durch das andere Thema dominiert sein könnte. Um dies zu verhindern, kann man – in einer erweiteten Variante – in jedem Iterationsschritt thematische Relevanzwerte (z.B. tf*idf-basierte Ähnlichkeiten zur ursprünglichen Anfrage) als multiplikative Gewichte der Vorgänger- bzw. Nachfolger-Gewichte einbauen.

Das HITS-Verfahren kann auch benutzt werden, um zu einer gegebenen Web-Seite ähnliche Web-Seiten zu ermitteln; dabei bezieht sich Ähnlichkeit auf die Linkstruktur, z.B. wären zwei Seiten sehr ähnlich, wenn sie genau dieselben Vorgänger und Nachfolger hätten. Für diese Art der Ähnlichkeitsanalyse bestimmt man zu einer gegebenen Web-Seite zunächst alle Nachfolger, alle oder eine beschränkte Anzahl der Vorgänger sowie (eine beschränkte Zahl von) Vorgänger(n) der Nachfolger und Nachfolger(n) der Vorgänger. Auf dieser Basismenge führt man dann den HITS-Algorithmus aus, und die ermittelten Autoritätsgewichte liefern eine Rangliste ähnlicher Seiten.

15.5 Automatische Klassifikation von Dokumenten

In bestimmten Anwendungen möchte man Dokumente automatisch klassifizieren, also aufgrund ihrer Featurevektoren bestimmten Themen (Klassen, Kategorien) zuordnen. Wenn man von einigen Trainingsdokumenten d1, ..., dn die Klassenzuordnung $c(di) \in \{C1, ..., Ck\}$ a priori kennt, weil diese Dokumente intellektuell klassifiziert worden sind, kann man weitere Dokumente mit a priori unbekannter Klasse durch statistische Ähnlichkeit mit den Trainingsdaten der verschiedenen Klassen mit einer bestimmten Wahrscheinlichkeit einer Klasse zuordnen. Beispielsweise kann man Dokumente, in denen die Wörter Theorem und Homomorphismus mit hohem (tf*idf-) Gewicht auftreten, mit hoher Wahrscheinlichkeit der Klasse Mathematik zuordnen, während bei Dokumenten mit hoch gewichteten Termen Tor, Elfmeter und Schiedsrichter vieles für die Klasse Sport spricht. Verfahren dieser Art gehören zur Familie des *Supervised Learning*, also der *Lernverfahren mit Trainingsdaten*. Wenn man keinerlei Trainingsdaten hat, kann man Verfahren des *Unsupervised Learning* anwenden, insbesondere die statistischen Verfahren der *Cluster-Analyse*. Im folgenden wird nur Klassifikation mit Trainingsdaten betrachtet.

Gegeben sind n Trainingsdokumente d1, ..., $dn \in [0,1]^m$ über einem m-dimensionalen Featureraum mit bekannten Klassen $c(di) \in \{C1, ..., Ck\}$ aus einer Menge von k verschiedenen Themen. Ein Klassifikator ist eine Funktion $c: [0,1]^m \rightarrow \{C1, ..., Ck\}$.

Automatische Klassifikation ist in IRS-Anwendungen auf vielfältige Weise nützlich:
- *Filtern:* teste eintreffende Dokumente (z.B. Mail, News), ob sie in eine interessante Klasse fallen
- *Übersicht:* organisiere Query-/Crawler-Resultate, Verzeichnisse, Feeds, etc.
- *Query-Expansion:* ordne Query einer Klasse zu und ergänze dementsprechende Suchterme
- *Relevanz-Feedback:* klassifiziere Treffer und lasse Benutzer relevante Klassen identifizieren, um bessere Query zu generieren
- *Query-Effizienz:* beschränke (Index-)Suche auf relevante Klasse(n)

kNN-Klassifikator

Der einfachste Klassifikator ist das sog. k-Nearest-Neighbor-Verfahren, kurz kNN. Es bestimmt zu einem zu klassifizierenden Dokument zunächst die k nächsten Nachbarn unter den Trainingsdaten gemäß einer Ähnlichkeitsfunktion über dem zugrundeliegenden Featureraum, z.B. der Cosinus-Ähnlichkeit. Dann ermittelt das Verfahren die Klassen dieser k nächsten Nachbarn und ordnet schließlich das neue Dokument der am häufigsten auftretenden Klasse zu. Beim letzten Schritt können die Klassenhäufigkeiten mit den Abständen der entsprechenden Trainingsdokumente zu dem neuen Dokument gewichtet werden.

Ordne d derjenigen Klasse Cj zu für die

$$f(\vec{d},C_j) = \sum_{\vec{v}\in kNN(\vec{d})} sim(\vec{d},\vec{v}) * \begin{cases} 1 & falls\ \vec{v}\in C_j \\ 0 & sonst \end{cases}$$

maximal ist.

Falls man nur binär klassifiziert, also nur testen will, ob ein Dokument zu einem gegebenen Thema passt oder nicht, ordnet man d zu, wenn $f(\vec{d},C_j)$ einen bestimmten Schwellwert δ überschreitet (z.B. δ=0.5).

Klassifikator nach Rocchio

Schritt 1:

Repräsentiere die Trainingsdokumente einer Klasse Cj

- mit tf*idf-Vektorkomponenten – durch den **Prototypvektor**:

$$\vec{c}_j := \alpha \frac{1}{|C_j|} \sum_{\vec{d}\in C_j} \frac{\vec{d}}{\|\vec{d}\|} - \beta \frac{1}{|D-C_j|} \sum_{\vec{d}\in D-C_j} \frac{\vec{d}}{\|\vec{d}\|}$$

mit geeigneten Koeffizienten α und β (z.B. α=16, β=4).

Schritt 2:

Ordne ein neues Dokument d derjenigen Klasse C_j zu, für die Cosinus-Ähnlichkeit cos(d,c_j) maximal ist.

Satz:

Für α=β=1 maximiert cj die Funktion:

$$f(\vec{c}_j) = \alpha \frac{1}{|C_j|} \sum_{\vec{d}\in C_j} cos(\vec{c}_j,\vec{d}) - \frac{1}{|D-C_j|} \sum_{\vec{d}\in D-C_j} cos(\vec{c}_j,\vec{d})$$

Naive-Bayes-Klassifikator

Bei diesem Verfahren schätzt man die bedingte Wahrscheinlichkeit, dass ein Dokument zur Klasse Cj gehört unter der Vorbedingung, dass es einen bestimmten Featurevektor hat. Durch den Satz von Bayes lässt sich dies zurückführen auf die Wahrscheinlichkeit, dass das Dokument diesen Featurevektor hat, wenn es zur Klasse Cj gehört. Diese Wahrscheinlichkeit lässt sich anhand der Trainingsdaten schätzen.

Einfache Variante: binäre Features

Bei dieser Variante wird nur das Vorkommen oder Nichtvorkommen eines Terms im Dokument berücksichtigt; die Häufigkeit bleibt unberücksichtigt. Der Featurevektor des Dokuments di ist also ein binärer Vektor (der Dimension |F|=m), den wir zur besseren Abgrenzung mit Xi bezeichnen.

Schätze $P[\, d \in c_k \,|\, d \;hat\; \vec{X}\,] = \dfrac{P[\, d \;hat\; \vec{X} \,|\, d \in c_k\,]\, P[\, d \in c_k\,]}{P[\, d \;hat\; \vec{X}\,]}$

$\sim P[\, X \,|\, d \in c_k\,]\, P[\, d \in c_k\,]$

$= \Pi_{i=1}^{m} P[\, X_i \,|\, d \in c_k\,]\, P[\, d \in c_k\,]$ unter der Annahme, dass die Features paarweise unabhängig sind bzw. der Linked-Independence-Annahme

$$\frac{P[X \mid d \in c_k]}{P[X \mid d \notin c_k]} = \prod_i \frac{P[X_i \mid d \in c_k]}{P[X_i \mid d \notin c_k]}$$

$= \Pi_{i=1}^{m} p_{ik}^{\;Xi} (1 - p_{ik})^{1-Xi}\; p_k$ mit empirisch - anhand der Trainingsdaten - zu schätzenden $p_{ik}=P[X_i=1|c_k]$, $p_k=P[c_k]$

Die Unabhängigkeitsannahme für Features ist eigentlich realitätsfern; sie erklärt das Attribut „naiv" im Namen des Verfahrens. Da es bei der Klassifikation aber nur auf eine relative Schätzung der Zugehörigkeitswahrscheinlichkeiten zu den verschiedenen Klassen ankommt, funktioniert das Verfahren trotz dieser groben Annahme erstaunlich gut.

Bessere Variante: Bag-of-Words-Modell

Hier werden die Häufigkeiten der Terme im Dokument berücksichtigt, nicht jedoch deren Positionen zueinander (daher Bag-of-Words). Man postuliert ein generierendes Modell, nach dem Dokumente erzeugt werden, und schätzt die Parameter dieses Modells anhand der Trainingsdaten. Eine simple Variante nimmt dazu an, dass man für jeden Term separat „würfelt" – mit einem zweiseitigen Würfel bzw. einer Münze, ob er auf Wortposition 1, 2, usw. erscheint; dies führt auf eine Binomialverteilung für die Termhäufigkeit. Eine präzisere Variante berücksichtigt die Nebenbedingung, dass die Summe aller Termhäufigkeit gleich der Dokumentlänge sein muss (nach Elimination von Stoppwörtern). Man „würfelt" dabei für jede Wortposition – quasi mit einem m-seitigen Würfel -, welcher Term dort erscheint. Dies führt auf eine sog. Multinomialverteilung. Die Wahrscheinlichkeit der verschiedenen Würfelseiten sind die Parameter dieser Verteilung und werden anhand der Trainingsdaten geschätzt.

Schätze $P[\, d \in c_k \,|\, d \;hat\; \vec{f}\,] \sim P[\, \vec{f} \,|\, d \in c_k\,]\, P[\, d \in c_k\,]$ mit Termhäufigkeitsvektor f

$= \Pi_{i=1}^{m} P[\, f_i \,|\, d \in c_k\,]\, P[\, d \in c_k\,]$ bei Featureunabhängigkeit

$= \Pi_{i=1}^{m} \binom{length(d)}{f_i} p_{ik}^{\;f_i} (1 - p_{ik})^{length(d) - f_i}\; p_k$ mit Binomialverteilung

bzw. präziser:

$= \binom{length(d)}{f_1\; f_2 \ldots f_m} p_{1k}^{\;f_1}\; p_{2k}^{\;f_2} \ldots p_{mk}^{\;f_m}\; p_k$ mit Multinomialverteilung und der Restriktion $\sum_{i=1}^{m} f_i = length(d)$

mit den Multinomialkoeffizienten $\binom{n}{k_1\, k_2 \ldots k_m} := \dfrac{n!}{k_1!\, k_2! \ldots k_m!}$

Beispiel für Naive-Bayes-Klassifikation mit Bag-of-Words-Modell

3 Klassen: c1 – Algebra, c2 – Analysis, c3 – Stochastik
8 Terme, 6 Trainingsdokumente d1, ..., d6: je 2 in jeder Klasse
$\Rightarrow$ p1=2/6, p2=2/6, p3=2/6

	Gruppe	Homomorphismus	Vektor	Integral	Limes	Varianz	Wahrscheinlichkeit	Würfel
	f1	f2	f3	f4	f5	f6	f7	f8
d1:	3	2	0	0	0	0	0	1
d2:	1	2	3	0	0	0	0	0
d3:	0	0	0	3	3	0	0	0
d4:	0	0	1	2	2	0	1	0
d5:	0	0	0	1	1	2	2	0
d6:	1	0	1	0	0	0	2	2

	Algebra	Analysis	Stochastik
	k=1	k=2	k=3
p1k	4/12	0	1/12
p2k	4/12	0	0
p3k	3/12	1/12	1/12
p4k	0	5/12	1/12
p5k	0	5/12	1/12
p6k	0	0	2/12
p7k	0	1/12	4/12
p8k	1/12	0	2/12

Klassifikation von d7: (0 0 1 2 0 0 3 0)

$$P[\vec{f}\,|d\in c_k]\,P[d\in c_k] = \binom{length(d)}{f_1\, f_2 \cdots f_m} p_{1k}^{f_1}\, p_{2k}^{f_2} \cdots p_{mk}^{f_m}\; p_k$$

für k=1 (Algebra): $= \binom{6}{1\ 2\ 3}\left(\frac{3}{12}\right)^1 0^2\, 0^3\ \ \frac{2}{6} \qquad = 0$

für k=2 (Analysis): $= \binom{6}{1\ 2\ 3}\left(\frac{1}{12}\right)^1\left(\frac{5}{12}\right)^2\left(\frac{1}{12}\right)^3\ \ \frac{2}{6} \qquad = 20 * \frac{25}{12^6}$

für k=3 (Stochastik): $= \binom{6}{1\ 2\ 3}\left(\frac{1}{12}\right)^1\left(\frac{1}{12}\right)^2\left(\frac{4}{12}\right)^3\ \ \frac{2}{6} \qquad = 20 * \frac{64}{12^6}$

Resultat: Ordne d7 der Klasse C3 (Stochastik) zu

Feature-Selektion

Bei der Klassifikation von Textdokumenten kann die Dimensionalität des Featureraums sehr groß sein, und man möchte aus Effizienzgründen lieber mit einem Raum niedrigerer Dimensionalität arbeiten. Außerdem möchte man das inhärente Rauschen im Featureraum unterdrücken, also nur die wichtigsten, für die jeweiligen Klassen charakteristischen, Terme berücksichtigen. Dies kann man mit Hilfe informationstheoretischer Maße realisieren, z.B. der **Kreuzentropie** zwischen Termen und Klassen, die in der Literatur auch als **Mutual-Information-Maß (MI-Maß)** bezeichnet wird:

$$MI(X_i, c_j) = \sum_{X\in\{X_i, \bar{X}_i\}} \sum_{C\in\{c_j, \bar{c}_j\}} P[X \wedge C]\ log \frac{P[X \wedge C]}{P[X]\,P[C]}$$

Dies ist ein Sonderfall der sog. **Kullback-Leibler-Distanz**, ein Maß für die Unterschiedlichkeit zweier Wahrscheinlichkeitsverteilungen, insbesondere zwischen einer zweidimensionalen Verteilung von Term und Klasse und einer Verteilung, bei der Term und Klasse unabhängig voneinander sind. Unabhängigkeit würde bedeuten, dass der Term in allen Klassen gleichwahrscheinlich ist, so dass er dann für die Klassifikation wenig geeignet wäre. Man wählt daher bei der Feature-Selektion gerade diejenigen m' << m Terme für eine Klasse aus, deren Kreuzentropie bzw. Kullback-Leibler-Divergenz am größten ist. Man beachte, dass diese Auswahl pro Klasse getroffen werden kann. Wenn man für alle Klassen denselben reduzierten Featureraum verwenden will, wählt man diejenigen Features mit den größten Werten der mit den Klassenhäufigkeiten gewichteten MI-Maße:

$$MI(X_i) = \sum_{j=1}^{k} P[c_j]\, MI(X_i, c_j)$$

Ergänzende Literatur

- C.D. Manning, H. Schütze: Foundations of Statistical Natural Language Processing, MIT Press, 1999.
- R. Baeza-Yates, B. Ribeiro-Neto: Modern Information Retrieval, Addison-Wesley, 1999.
- N. Fuhr: Information Retrieval, Skriptum zur Vorlesung im WS 00/01, Universität Dortmund, http://ls6-www.informatik.uni-dortmund.de/ir/teaching/lectures/ir_ws00-01/irskall.pdf
- G. Weikum: Information Retrieval, Unterlagen zur Vorlesung im WS 00/01, Universität des Saarlandes, http://www-dbs.cs.uni-sb.de/lehre/ir-ws00/index.normal.html
- J. Han, M. Kamber: Data Mining - Concepts and Techniques, Morgan Kaufmann, 2001
- L. Gravano (Editor): IEEE CS Data Engineering Bulletin Vol.23 No.3, Special Issue on Next Generation Web Search, September 2000
- S. Sarawagi (Editor): IEEE CS Data Engineering Bulletin Vol.24 No.3, Special Issue on Imprecise Queries, September 2001
- L. Gravano (Editor): IEEE CS Data Engineering Bulletin Vol.24 No.4, Special Issue on Text and Databases, December 2001
- G. Weikum (Editor): IEEE CS Data Engineering Bulletin Vol.25 No.1, Special Issue on Organizing and Discovering the Semantic Web, March 2002S. Brin, L. Page: The Anatomy of a Large-Scale Hypertextual jWeb Search Engine, WWW Conference 1998.
- J.M. Kleinberg: Authoritative Sources in a Hyperlinked Environment, Journal of the ACM Vol.46 No.5, 1999