9. Text- und Web-Mining

Text Mining:
Anwendung von Data Mining - Verfahren auf große Mengen von Online-Textdokumenten

Web Mining:
Anwendung von Data Mining - Verfahren auf Dokumente aus dem WWW oder auf zugehörige Informationen (Link-Struktur, Benutzung der Dokumente)

Besonderheiten von Textdokumenten:

- unstrukturiert
- Probleme des automatischen Textverstehens (vage Inhalte, Metaphern-Gebrauch, gleiches Wort kann verschiedene Bedeutungen haben...)
- Inkonsistenzen möglich
- verschiedene Sprachen

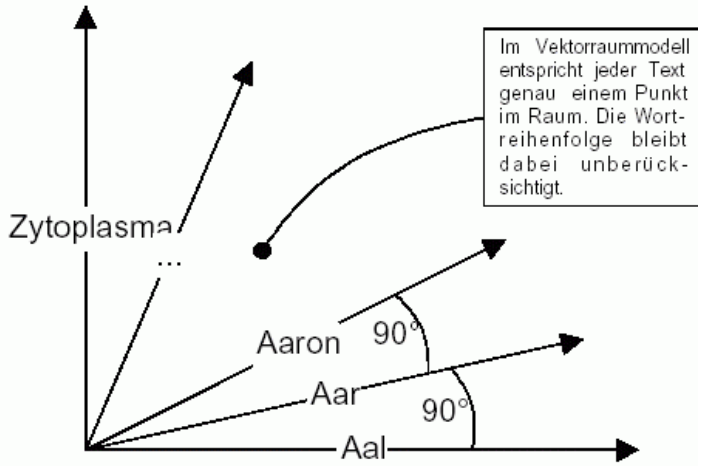
$\rightarrow$ generiere aus Textdokumenten zunächst Metadaten, wende Data Mining - Techniken dann nur auf diese an

2 Ansätze:

- Extraktion von Merkmalen für jedes Dokument, die den Inhalt grob charakterisieren sollen (Merkmalsauswahl manuell oder automatisch, oft "Wörterbücher") – Beisp.: Namen, Organisationen, Orte, Zeitangaben, Fachbegriffe

- Transformation jedes Dokuments in einen hochdimensionalen Vektor von Term-Häufigkeiten (Term: 1 oder mehrere aufeinanderfolgende Wörter)

Vektorraum-Modell



- Text wird repräsentiert durch Punkt im hochdimensionalen Raum,
- Wortreihenfolge bleibt unberücksichtigt

aus Scheffer & Bickel 2004

(vgl. Kap. 6, Bayes-Klassifikation)

Dimension der Vektoren = Anzahl der relevanten Terme (bis zu 100 000 und mehr!)

relevante Terme:

- manuelle Auswahl

oder

- automatische Bestimmung

zur Reduktion der Anzahl der Terme spezielle Techniken:

- *Stop-Listen* = Wörter, die ignoriert werden ("und", "der", "HTML" usw.)
- *Stemming* = Reduzierung der Wörter auf eine Grundform ("Ängste" → "Angst")
- Entfernen von *besonders häufigen* und *besonders seltenen* Termen

Beim Web-Mining Verwendung zusätzlicher Struktur-informationen aus HTML-Tags und Metadaten der Webseiten

Umgang mit Termhäufigkeiten:

- Termfrequenz eines Wortes in einem Text = # Vorkommen des Wortes im Text.
- Problem: Lange Texte haben lange Vektoren, führt zu Verzerrungen beim Ähnlichkeitsmaß.
- Problem: Einige Wörter sind weniger relevant (und, oder, nicht, wenn, …)
- Lösung: Inverse Dokumentenfrequenz

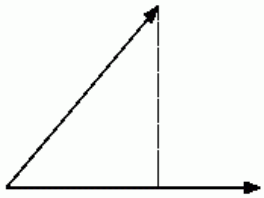
$$IDF(wort_i) = \log \frac{\#\text{Dokumente}}{\#\text{Dokumente , in denen Wort}_i \text{ vorkommt}}$$

- TF-IDF ($Wort_i$) = TF($Wort_i$) * IDF($Wort_i$).
- Vektor wird normiert.
- Repräsentation eines Textes:

$$TFIDF(Text) = norm \begin{pmatrix} TF(Wort_1) \cdot IDF(Wort_1) \\ \vdots \\ TF(Wort_n) \cdot IDF(Wort_n) \end{pmatrix}$$

Ähnlichkeitsmaß für die Clusteranalyse und Klassifikation:

- Ähnlichkeit zwischen Texten x_1 und x_2:
- Cosinus des Winkels zwischen den Vektoren.



- Ähnlichkeit: $sim(x_1, x_2) = \cos(\theta) = \frac{x_1 \cdot x_2}{|x_1| \cdot |x_2|}$
- Zwischen 0 und 1.

Anwendung der Data Mining - Techniken auf diese Daten:

Beispiele für Anwendungen von Clusteringverfahren

- Clustering von Dokumenten im allgemeinen, um einen groben Überblick über eine Menge von Dokumenten zu bekommen, oder um ähnliche und damit in Beziehung stehende Informationen in der Menge von Dokumenten zu bestimmen.
- Clustering von Antwortmengen einer Suchmaschine zur intelligenten und zusammenfassenden Präsentation von im allgemeinen sehr großen Mengen von Suchergebnissen, etwa durch geeignete Beschreibung der vorhandenen Cluster von Dokumenten.
- Clustering von Webdokumenten unter Berücksichtigung der Link-Struktur, um etwa sogenannte Communities zu finden.
- Clustering von Web-Sessions zur Bestimmung von Benutzergruppen einer Web-Site (siehe auch Abschnitt 3.1.4).

Beispiele für Anwendungen von Klassifikationsverfahren

- Automatische Zuordnung von Dokumenten in eine vorgegebene Taxonomie, z.B. thematische Gruppierung und Speicherung in vorgegebenen Verzeichnissen.
- Automatische Zuordnung eingehender Mails zu bestimmten Abteilungen/Mitarbeitern einer Firma.
- Klassifikation von Antworten einer Suchmaschine aufgrund von User Feedback auf vorhergehende Anfragen bzw. auf schon gelieferte Antworten zur aktuellen Anfrage.
- Klassifikation von Web-Sites beispielsweise als potentielle Kunden oder Konkurrenten.
- Klassifikation von Links auf einer Internetseite z.B. als interessant oder relevant für eine Anfrage.
- Klassifikation von e-mails als Spam / Nicht-Spam
- thematische Klassifikation von e-mails

Beispiele für Anwendungen von Assoziationsregeln

- Finden von Zusammenhängen und Abhängigkeiten zwischen Mengen von Begriffen, die in einer Sammlung von Dokumenten vorkommen.
- Finden von häufigen Zugriffsmustern, das heißt Reihenfolgen, in denen auf die Internetseiten einer Web-Site zugegriffen wird.

Beispiele für speziellere Techniken

- Entdecken von Veränderungen der Begriffs- oder Konzeptverteilungen in Zeitungsmeldungen zu einem bestimmten Thema. Dies kann dazu dienen, unerwartete oder interessante Trends und Abweichungen von den „normalen" Beziehungen zwischen politischen Akteuren zu erkennen.
- Analyse wechselseitiger Zitierung in wissenschaftlichen Veröffentlichungen.
- Analyse von Veröffentlichungen über Forschungsergebnisse aus verschiedenen Bereichen, die zwar einen ähnlichen Sachverhalt betreffen, die sich jedoch gegenseitig nicht zitieren. Dies kann zu neuen und richtigen Hypothesen über diese Sachverhalte führen, die bisher noch keinem der Fachwissenschaftler aufgefallen sind, weil diese im allgemeinen gar nicht die Zeit haben, auch noch die Literatur zu lesen, die in einem anderen als ihrem eigenen Fachbereich erschienen ist. Ein prominentes Beispiel für einen solchen Erfolg ist die Entdeckung des Zusammenhangs zwischen Magnesiummangel und Migräne (siehe z.B. [Swanson & Smalheiser 1994] für Details).

(modifiziert nach Ester & Sander 2000)

Beispiel "Klassifikation"

Textklassifikation

- Textklassifikator: Ordnet einen Text einer Menge von inhaltlichen Kategorien zu.
- Klassifikator wird aus Sammlung klassifizierter Beispieltexte gelernt.
  - Lineare Klassifikation.
- Anwendungsbeispiele:
  - Einordnung von Webseiten in Web-Directory,
  - Klassifikation von Zeitungsmeldungen,
  - Erkennung von Spam-Emails.

oft benutzt: Naive Bayes-Klassifikation (vgl. Kap. 6)

- Verwendet *keine* TDIDF-Repräsentation sondern berücksichtigt nur, welche Wörter vorkommen.
- Um die Fehlklassifikationswahrscheinlichkeit gegeben x zu minimieren, gib für jede Instanz x die wahrscheinlichste Klasse y zurück. (Instanzen x sind Wörter (x_1, ..., x_n)).
- Unabhängigkeitsannahme für Attribute.

$$\begin{aligned}
&\max_y P(y \mid x_1,...,x_n)) \\
&= \max_y P((x_1,...,x_n) \mid y)P(y) \\
&= \max_y P(x_1 \mid y)...P(x_n \mid x_1,...,x_{n-1},y)P(y) \\
&\approx \max_y P(x_1 \mid y)...P(x_n \mid y)P(y)
\end{aligned}$$

- Leider sind die Wahrscheinlichkeiten P(y) und P(x_i | y) unbekannt.
- Klassifikation ist nur dann optimal, wenn Attribute unabhängig sind und Wahrscheinlichkeiten bekannt sind.
- Wir können sie immerhin schätzen, z.B.

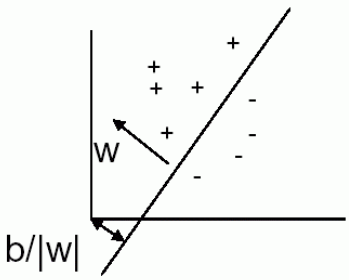
$$P(x_i \mid y) = \frac{\#\text{Beispiele Klasse y mit Wort } x_i}{\#\text{Beispiele Klasse } y}$$

$$P(y) = \frac{\#\text{Beispiele Klasse } y}{\#\text{Beispiele}}$$

Alternative: "Lineare Klassifikatoren" –
Trennung durch Hyperebenen

Lineare Klassifikatoren

- Positive und negative Beispiele einer Klasse ergeben jeweils eine Punktwolke im Merkmalsraum.
- Gesucht: Trennebene, die die Punktwolken separiert.



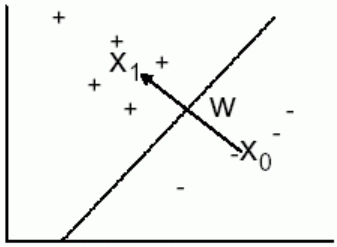
$$f(x) = wx + b$$

Punkte auf der Geraden: f(x)=0

$$h(x) = sign(f(x))$$

→ Verfahren von Rocchio

Rocchio



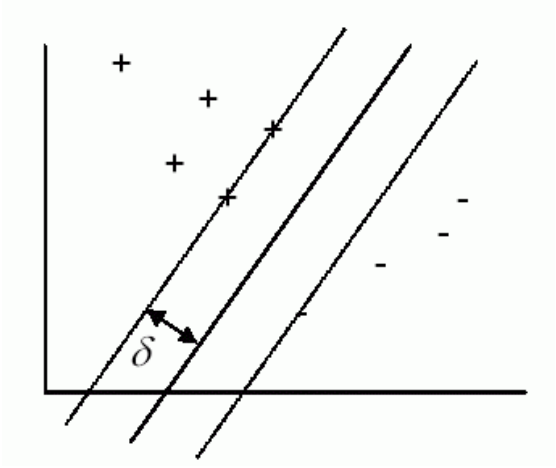
- X_0: Mittelpunkt der neg. Beispiele
- X_1: Mittelpunkt der pos. Beispiele
- Trennebene: Normalenvektor = (x_1-x_0)

$$f(x) = d(x_1, x) - d(x_0, x)$$
$$= (x_1 - x_0)x + b$$
$$h(x) = sign(f(x))$$

- Achsenabstand: $(x_1-x_0)/2$ muss auf der Ebene liegen.

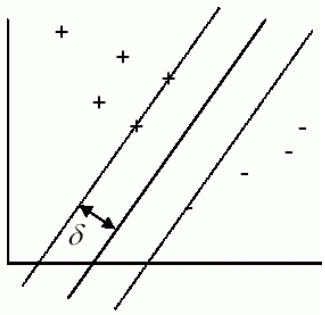
Trennebene hat maximalen Abstand von den Mittelpunkten (Zentroiden) der Klassen
hat sich in der Praxis der Textklassifikation besser bewährt als die naive Bayes-Klassifikation
aber: Möglichkeit, dass Trainingsbeispiele falsch klassifiziert werden

Verbesserung des Verfahrens: Trennebene soll maximalen Abstand zwischen sich selbst und den (einzelnen) Beispielen annehmen
→ "Large-Margin-Klassifikatoren"



Support-Vektor-Maschinen

- Für alle Punkte: $y_i \left(\frac{w}{|w|} x_i + \frac{b}{|w|} \right) \geq \delta$
- Mit möglichst großem δ
- Äquivalent: $y_i \left(w x_i + b \right) \geq 1$
- Mit minimalem |w|.



- Duale Repräsentation: Ebene ist Lösung des folgenden Optimierungsproblems:
  - Minimiere $\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$
- Quadratisches Optimierungsproblem; Funktion ist konvex (genau ein Optimum), lösbar in n^2.
- $\alpha_i > 0$, wenn (x_i, y_i) minimalen Abstand zur Ebene hat („Support-Vektoren“).

(aus Scheffer & Bickel 2004)

- Aufwand beim Lernen: quadratisch in der Anzahl der Beispiele
- Aufwand beim Anwenden: linear in der Anzahl der Support-Vektoren

Anwendungsgebiete: neben Textklassifikation auch Handschrifterkennung, Bildverarbeitung, Klassifikation von DNA-Sequenzen

Beispiel "Clusteranalyse"

Text-Clustering

- agglomerative Verfahren: prinzipiell für Texte möglich, aber Probleme bei der Skalierung – zu große Distanzmatrizen

- k-means-Verfahren: gut geeignet für Texte, etwas weniger stabil, aber sehr effizient

- Erwartungsmaximierungs-Verfahren mit Normalverteilungsannahme: nicht gut geeignet, da längennormierte TFIDF-Vektoren nicht normalverteilt sind

→ Anpassung des EM-Verfahrens:

- Multinomiale Mischverteilung.
- Modell: Text x aus einer Mischkomponente y wird generiert indem *m* Wörter zufällig mit Zurücklegen gezogen werden.
- Wahrscheinlichkeit dafür, dass Wort w_1 x_1- mal, …, und w_n x_n-mal gezogen wird ist multinomialverteilt:

$$P(x_1,...,x_n) = \frac{m!}{x_1!...x_n!} P(x_1 \mid y)^{x_1}...P(x_n \mid y)^{x_n}$$

- Kann in EM-Algorithmus eingebaut werden.
- Berücksichtigt den IDF-Wert nicht.

noch besser angepasstes Clustering-Verfahren:
Suffix-Tree-Clustering

- berücksichtigt bei Ähnlichkeitsbestimmung das gemeinsame Auftreten ganzer Satzteile in den Dokumenten
- Anwendung: Clustering von Web-Suchergebnissen
- Verfahren auch anwendbar auf "Snippets" (das, was Google für jedes Suchergebnis ausgibt...)
- Laufzeit nur linear abh. von Anzahl der Suchergebnisse

Algorithmus STC (Suffix Tree Clustering)

1. Datenvorbereitung
- Anwendung eines Stemming-Algorithmus (Plural in Singular, Präfixe und Suffixe entfernen...)
- HTML-Tags, Zahlen und Satzzeichen werden entfernt
- Satzanfänge und -enden werden markiert
- von jedem Wort werden Zeiger auf die Position im Originaldokument gespeichert

2. Identifikation von Basisclustern
- Basiscluster: Menge von Dokumenten, die eine Phrase (= geordnete Sequenz von Wörtern) gemeinsam haben

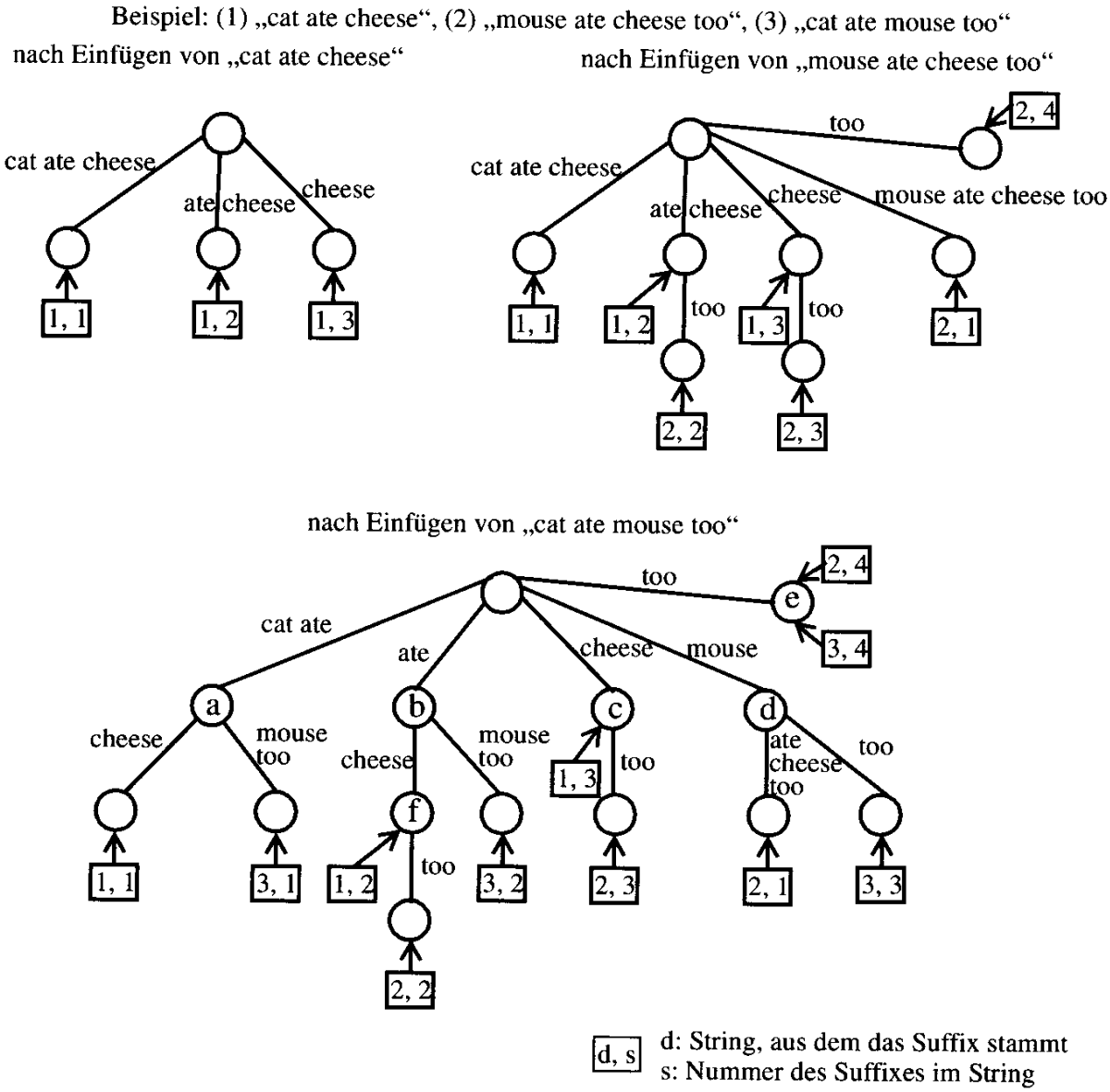
(Basiscluster sind nicht notw. disjunkt)

Auffinden der Basiscluster durch Aufbau eines *Suffix-Baumes* für alle Sätze in der gesamten Dokument-Menge

Suffix-Baum einer String-Menge S:
- jeder innere Knoten hat ≥ 2 Tochterknoten
- jede Kante ist mit einem nichtleeren Teilstring aus S beschriftet
- Beschriftung eines Knotens *x* = Konkatenation aller Kantenbeschriftungen auf dem Pfad Wurzel$\rightarrow$*x*
- Beschriftungen aller von einem Knoten ausgehenden Kanten beginnen mit verschiedenen Wörtern
- für jedes Suffix s eines jeden Strings aus S existiert ein Knoten im Baum mit Beschriftung s
- in jedem Knoten wird zusätzlich die Nummer des Herkunftsstrings seiner Beschriftung und die Positionsangabe des Suffixes im Herkunftsstring gespeichert

Konstruktion des Suffix-Baumes durch inkrementelles Erweitern

Beispiel (aus Ester & Sander 2000):



Jeder Knoten im Suffixbaum repräsentiert einen Basiscluster (d.h. eine Menge von Dokumenten und eine Phrase, die diesen Dokumenten gemeinsam ist)

im obigen Beispiel – mit a bis f markierte Knoten:

Knoten	Phrase	Dokumente (Strings)
a	„cat ate“	1, 3
b	„ate“	1, 2, 3
c	„cheese“	1, 2
d	„mouse“	2, 3
e	„too“	2, 3
f	„ate cheese“	1, 2

Wichtigkeit eines einzelnen Basisclusters:
Anzahl der Dokumente im Cluster · Anzahl der Wörter in der Phrase

3. Kombination von Basisclustern

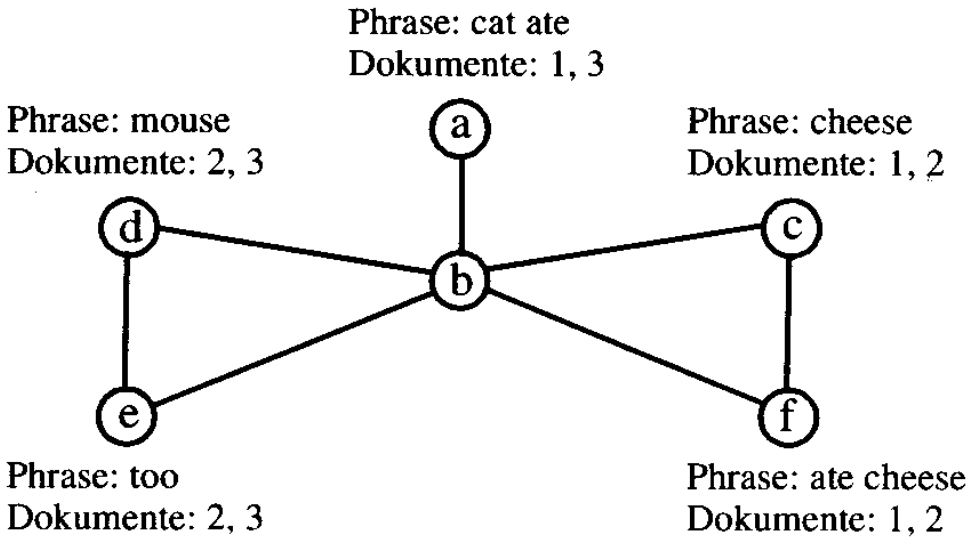
stark überlappende Basiscluster werden verschmolzen

- dazu binäres Ähnlichkeitsmaß zwischen 2 Basisclustern A und B:

Wert 1, wenn $|A \cap B|/|A| > 0{,}5$ und $|A \cap B|/|B| > 0{,}5$
Wert 0 sonst

→ Graph auf den Basisclustern (Kante, wenn Ähnlichkeitsmaß den Wert 1 hat)
Cluster = Zusammenhangskomponente in diesem Graphen

im Beispiel nur 1 Komponente (nur 1 Cluster):



diese Cluster können dann benutzt werden, um z.B. Webseiten oder Suchergebnisse zu klassifizieren.

Informationsextraktion

- Named Entity Recognition: Erkennung von Eigennamen, Maßzahlen, Datumsangaben usw.
- Informationsextraktion: Erkennung von Phrasen (häufig Named Entities), die eine bestimmte Bedeutung tragen. Z.B.:
  - Abflugdatum, Ankunftsdatum,
  - Anzahl von Opfern eines Anschlages,
  - Zielort eines Staatsbesuches.
  - Erkennen der Art einer Relation zwischen Entitäten (Mitarbeiter-von, Standort-von, Produkt-von, …).
  - Interaktionsbeziehung von Genen.

Evaluation von Informationsextraktoren

- MUC – Message Understanding Conferences: Wettbewerbe bis 1996.
  - Artikel aus New York Times.
- BioCreative – Informationsextraktion aus biomedizinischen Veröffentlichungen, seit 2003.
- Precision = Korrekt gefüllte Slots / gefüllte Slots.
- Recall = Korrekt gefüllte Slots / Slots.
- F-Measure = 2(Precision*Recall)/(Precision+Recall).

Ansätze der Informationsextraktion aus Texten:

generische Ansätze:

- Tokenisierung
- morphologische Analyse der Texte
- Parsing
- Named Entity Recognition (z.B. auf Grundlage eines Dictionary...)

problemspezifisch:

- speziell für das Problem trainierte statistische Modelle oder SVMs, oder handgeschriebene Grammatiken

Web Mining

- Klassifikation des Web Mining nach benutzten Datentypen:
  - **Web Content**
    - Daten direkt aus Web-Seiten, z.B. HTML, Multimediadateien und Graphiken
  - **Web Structure**
    - Hyperlink-Struktur zwischen Web-Seiten
  - **Web Usage**
    - Beschreibung der Nutzung von Web-Ressourcen, z.B. Log-Dateien von Proxy-Servern und Web-Servern

Web Content Mining:

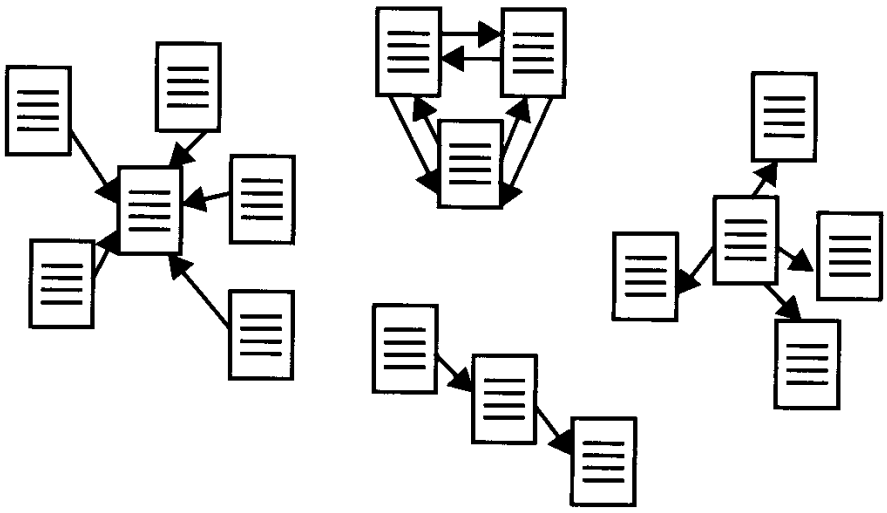
- Entdeckung nützlicher Informationen aus Web Content
- Zerlegung in zwei unterschiedliche Sichten
  - **Information-Retrieval-Sicht**
  - **Datenbanksicht**

- **Information-Retrieval-Sicht**
  - Unterstützung des Auffindens der Informationen oder der Filterung der Informationen
  - **Hauptdaten** → **Darstellung**
    (Hyper-)Textdokumente → Ausdrücke, Beziehungen, Wortmengen
  - **Applikation**:
    - Dokument-Kategorisierung
    - Auffinden der Muster in Texten
    - Auffinden der Extraktionsregeln

- **Datenbanksicht**
  - Modellierung der Daten für Netzanwendungen und Integration der Daten im Netz, um **komplizierte** Abfragen auszuführen
  - **Hauptdaten** → **Darstellung**
    Hypertext-Dokumente → Beziehungen, Edge-Labeled Graph
  - **Applikation**:
    - Auffinden von frequenten Substrukturen
    - Entdeckung von Web-Seiten-Schemata

Web Structure Mining:

Linkstruktur (Hypertext-Charakter des WWW) steht im Vordergrund



- Entdeckung des Modells der Verweisstruktur des Web
- Hauptdaten → Darstellung
  Linkstrukturen → Graphen
- Applikation
  - Generierung von Information über Unterschied und Ähnlichkeit zwischen den Web-Seiten
  - Maß der Vollständigkeit der Web-Seiten
  - Maß der Relevanz der Web-Seiten

Beispiel Google: Das Page Rank - Konzept

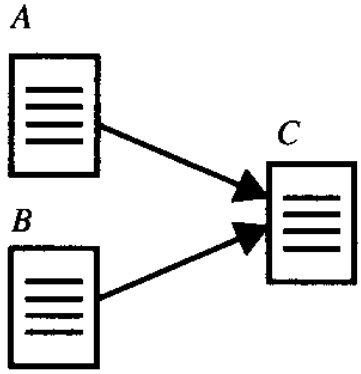
Der Algorithmus, mit dem Google seine Suchergebnisse sortiert, ist Firmengeheimnis und wird ständig angepasst

- Gefahr des Missbrauchs (Link-Farmen; "Google-washing" von Begriffen)

ein wichtiger Bestandteil von Google's Page Rank ist die auf der Linkstruktur beruhende, rekursive Definition von "Wichtigkeit" einer Seite:

> *Die Wichtigkeit einer Seite ist umso größer, je mehr wichtige Seiten auf sie verweisen*

Präzisierung:
Sei *FLinks*(*A*) die Menge aller ausgehenden Links einer Seite *A*, *BLinks*(*A*) die Menge aller eingehenden Links von *A*.



$$FLinks(A) = \{C\}$$
$$FLinks(B) = \{C\}$$
$$BLinks(C) = \{A, B\}$$

- eine Seite *A* hat hohen Page Rank, wenn die Summe der Page Ranks ihrer eingehenden Links hoch ist,
- eine Seite *B* verteilt ihre Wichtigkeit in gleichen Teilen auf alle Seiten, auf die sie verweist:

$$PageRank(A) = \frac{1}{c} \sum_{B \in BLinks(A)} \frac{PageRank(B)}{|FLinks(B)|}.$$

(*c* = Normierungsfaktor)

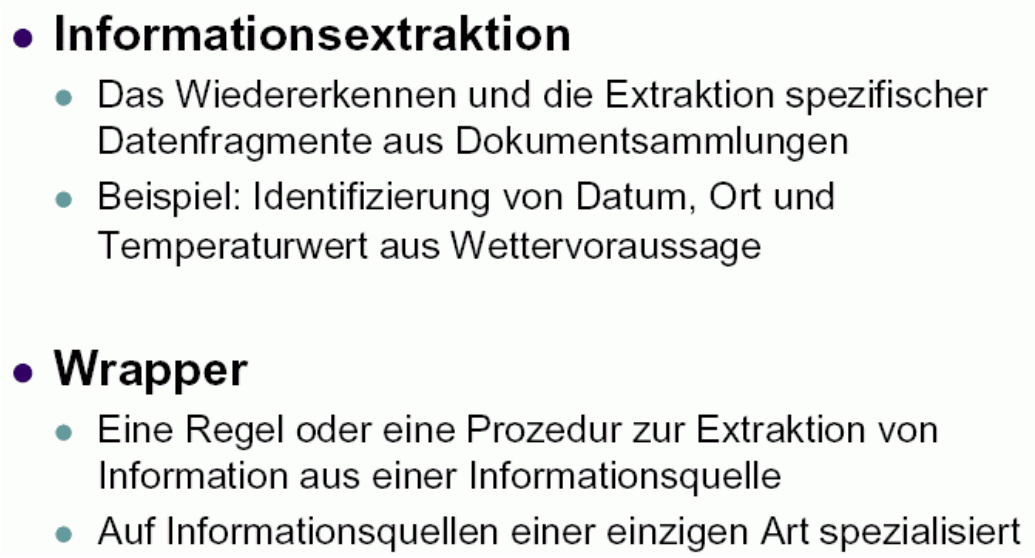
iterative Bestimmung des Page Rank:

- man nimmt eine beliebiege Zuweisung von Werten für alle Webseiten vor (typischerweise 1)
- wiederhole die Berechnung mittels der obigen Formel für alle Seiten so lange, bis die Werte stabil bleiben
- sie konvergieren gegen die Eigenvektoren der Adjazenzmatrix der Menge von Webseiten mit ihren Links

zusätzlich wird für den Google-Page Rank verwendet:

- räumliche Nähe der Suchbegriffe zueinander (Abstand im Text)
- Anchor Text der Links (Text, der zum Anklicken angezeigt wird): Eine Seite *A* bekommt höhere Wichtigkeit, wenn auch die Anchor-Texte von Links, die auf *A* verweisen, Suchbegriffe enthalten

Informations-Extraktion aus dem Web



Wrapper-Generierung:

- manuell (nach sorgfältiger Überprüfung von Beispiel-Webseiten), oder
- automatisch durch Lernen aus Beispielseiten (induktiv)

Web-Crawler:

durchsuchen automatisch das WWW nach bestimmten Inhalten

Motivation:
Suchergebnisse von Standard-Suchmaschinen

- oft nicht mehr aktuell,
- erfassen nur kleinen Anteil aller Webseiten,
- sind nicht problemspezifisch

Ablauf:

1. Manuelle Spezifikation der interessanten Themen anhand von Trainingsseiten
2. Interaktives Lernen von Klassifikationsregeln: Klassifikator lernt Regeln, die Webseiten als "interessant" oder "uninteressant" klassifiziert werden können. Benutzer kann den Klassifikator in dieser Phase noch interaktiv korrigieren.
3. Automatischer Crawl: Ausgehend von den Trainings-dokumenten folgt der Crawler der Linkstruktur. Alle gefundenen Seiten, die von Klassifikator als "interessant" eingestuft werden, werden als Antworten zurückgeliefert und als Ausgangspunkt für die weitere Suche genutzt. Keine Interaktion mit dem Benutzer.

2 Komponenten eines Crawlers:

- *Klassifikator*: Klassifikation meist mit angepasster Variante der Bayes-Klassifikation (s. Kap. 6)
- *Distiller*: Legt Reihenfolge fest, in der die von einer interessanten Seite ausgehenden Links verfolgt werden

gute Strategie für Crawler:
möglichst schnelles Auffinden von "*Hubs*" ("Naben") in der Linkstruktur

verschränkt rekursive Definition:

Eine gute *Authoritative Page* ist eine Seite, die von vielen guten Hubs referenziert wird
ein guter *Hub* ist eine Seite, die auf viele gute Authoritative Pages verweist.

Präzisierung:

$$HubRank(A) = \sum_{B \in FLinks(A)} AuthorityRank(B),$$

$$AuthorityRank(A) = \sum_{B \in BLinks(A)} HubRank(B).$$

iterative Berechnung:

- starte mit beliebiger initialer Zuweisung von Werten für jede Webseite (z.B. überall 1)
- iteriere Berechnung der obigen Formeln für jede Seite, bis Werte stabil bleiben

die Werte konvergieren gegen die Eigenvektoren der Matrix M^TM für den Authority Rank bzw. MM^T für den Hub Rank (M = Adjazenzmatrix der Link-Struktur)

Maß für Effektivität eines Crawlers:

Page Acquisition Rate (Durchschnittl. Häufigkeit gefundener relevanter Seiten je Zeiteinheit)
- kann auch nach Verfolgen vieler Links von den ursprünglichen Trainingsseiten noch sehr hoch sein!

(nach Ester & Sander 2000)

Web Usage Mining

- Entdeckung der Zugriffsmuster
- Sammlung der Informationen über Benutzerverhalten
- Hauptdaten → Darstellung
  Server/Browser Logs → Graphen, relationale Tabellen
- Applikation
  - Konstruktion von Web-Seiten
  - Marketing
  - Benutzer-Modellierung

Web Usage Mining: Mining Server Logs

- Web Usage Mining: Ziele.
  - Lernen über Nutzerverhalten.
  - Welche Seiten werden in denselben Sessions / von denselben Nutzern angesehen?
  - Gibt es Gruppen von Nutzern mit zueinander ähnlichem Verhalten?
  - Wo werden Sessions ohne Kauf abgebrochen?
- Server-Log-Einträge:
  - 123.456.78.9 [02/Dec/2002:16:44:27 –0500] „GET index.html HTTP/1.0“ 200 3290 – Mozilla/3.04 (Win98, I)
  - IP-Addresse, Datum, HTTP-Kommando, Status, Größe, Referrer, Browser,…

- Logs werden in Sessions unterteilt.
- Session = einzelner Besuch eines einzelnen Benutzers einer Website.
- Sequenzdaten:
  - Folge von Seiten einer Session.
  - Verarbeitung mit Sequenzmodellen (z.B. Hidden-Markov-Modellen).
- Vektordaten:
  - Menge von Seiten einer Session.
  - Verarbeitung mit Standardverfahren des maschinellen Lernens.

Welche Sessions verlaufen "erfolgreich" (werden z.B. mit einem Kauf abgeschlossen)?

- Klassifikationsproblem mit Sequenzdaten
- z.B. Hidden-Markov-Modelle (stochastische Prozesse mit diskretem Zustandsraum) oder Entscheidungsbäume

Gibt es Gruppen ähnlicher Nutzer?

- Clusteranalyse auf Vektor- oder Sequenzdaten
- z.B. Cluster der "skandalinteressierten Leser", die auf Politik-, Sport- und Wirtschaftsseiten nach Skandalmeldungen suchen (wurde von einem Verlag auf diese Weise identifiziert!)

- Welche Seiten werden in derselben Session angesehen?
  - Assoziationsregeln über Vektordaten.
  - Normaler Apriori-Algorithmus.
- Welche Sessions verlaufen erfolgreich?
  - Klassifikationsproblem.
  - Pos Bsp: Sessions mit Kauftransaktion. Neg: ohne.
  - Z.B. Entscheidungsbaumverfahren.
  - Interpretation des Baumes kann Hinweise zur Verbesserung der Seite liefern (z.B. wenn AGB gelesen dann keine Transaktion).

Lernen von Matchingfunktionen aus Server-Logs von Suchmaschinen

- Suchmaschinen-Logs: Anfragen, zurückgelieferte Seite, nächster Klick des Benutzers.
- Suchmaschinen verwenden Matchingfunktion, die Übereinstimmung von Anfrage und Seite beschreiben, z.B.:

$$\Phi(d,q) = \begin{pmatrix} \langle d,q \rangle \\ PageRank(d) \\ \#\,gleiche\ Wörter\ im\ Titel \\ \dots \end{pmatrix}$$

- Lineare Funktion gewichtet die einzelnen Matchattribute, Ranking nach Ergebnis.

$$F_q(d) = \langle w, \Phi(d,q) \rangle$$

- Was ist beste Rankingfunktion w?
- Idee: Lernen aus Server-Log. Wenn Benutzer nicht den ersten sondern den k-ten Link anklickt, dann bewertet er damit die ersten k-1 Seiten als schlechter als den k-ten.

(aus Scheffer & Bickel 2004)