Häufige Mengen

- Grundalgorithmen
  - Apriori
  - FP Growth
- Verbesserungen
  - Kondensierte Repräsentationen
  - Pushing Constraints into the algorithm
  - Bessere Signifikanztests

Häufige Mengen

- Erweiterungen zur Zeit
  - Episodenlernen mit WINEPI
  - Zeitreihen lernen nach Das
  - Zeitintervallbeziehungen lernen nach Hoeppner
- Privacy preserving data mining
  - K-anonymity (pushing constraints into FSM)
- Clustering anhand häufiger Mengen
  - → Ansatz von Ester
  - Ansatz von Strehl und Ghosh
  - ...

Clustering von Dokumenten

- Die Menge aller Wörter, die in irgend einem Dokument einer Sammlung D vorkommen, bilden die Menge der Terme T.
- Die Menge $F=\{F_1 F_2 ... F_k\}$ besteht aus allen Termmengen $F_i \subseteq T$,die häufiger als smin in der Sammlung vorkommen.
- Ein Dokument D_i ist die Menge der Terme, die in diesem Dokument vorkommen, $D_i \subseteq T$.
- Ein Dokument wird von einer häufigen Menge F_i abgedeckt, wenn alle Terme in F_i auch in dem Dokument vorkommen. $Cov(F_i) = \{D_i \in D \mid F_i \subseteq D_i\}$
- Ein cluster ist die Menge aller abgedeckten Dokumente $Cov(F_i)$ und die häufigen Terme F_i sind seine Beschreibung.
- Es soll eine Mege von clusters gefunden werden, die sich möglichst wenig überlappt.

Frequent Term-Based Clustering

- Datenbank von Dokumenten D
- Berechnen aller häufigen Mengen mit Apriori oder Fpgrowth
- Bei jeder häufigen Menge ist angegeben, welche Dokumente sie abdeckt (i.e. support ist hier cov)
- Aufgabe: wähle diejenigen häufigen Mengen, deren cover sich möglichst wenig überdeckt.

Beil, Ester, Xu (2002) Frequent Term-Based Text Clustering, in KDD 2002

Fung, Wang, Ester (2003) Hierarchical Document Clustering Using Frequent Itemsets, in SDM 2003

Overlap

- f: Anzahl der Termmengen, die ein Dokument unterstützt.
- Entropy Overlap eines clusters C
  $EO(C) = \sum_{D \in C} - 1/f_i \ln(1/f_i)$
- Wenn alle Dokumente von C keine anderes Termmenge unterstützen, ist
  $EO(C)=0$, $f_i=1$.

D1

{sun}, {fun}, {surf}, {sun,fun}, {fun, surf}, {sun, fun, surf}

$f_i = 6$

{sun} {beach}

D1, D4, D5, D6, D2, D9, D13 D8, D10, D11, D15

D7, D14 D2, D9, D13 D8, D10, D11, D15

{sun, fun, beach}

D8, D10, D11, D15

Algorithmus FTC

Gegeben: F häufige Mengen,
n=|D| Anzahl Dokumente,
select={ }
While |cov(select)|≠n do
For each C_i in F calculate $EO(C_i)$,
best:= C_i with minimal $EO(C_i)$
select:=select ∪ best
F:=F \best
D:=D \cov(best)
Return:
select,
$cov(C_i)$, $C_i \in$select

Eine Iteration:
F: {sun}, {fun}, {surf}, {beach}, {sun,fun}, {fun, surf}, {sun, beach}, {sun, fun, surf},{sun, fun, beach}

n=16

EO({sun})=2,98
EO({fun})=3,0
EO({beach})=2,85 ...
EO({sun,fun})=1,97
EO({sun,beach})=1,72...
EO({sun,fun,beach})=0,9

select:={sun,fun,beach}
F:= {sun}, {fun}, {surf}, {beach}, {sun,fun}, {fun, surf}, {sun, beach}, {sun, fun, surf}

...

Hierarchisches FTC

- FTC-Algorithmus nimmt nicht F, sondern nur häufige Mengen der Länge k als Eingabe
- Ein Cluster mit F_i der Länge k wird weiter aufgeteilt in cluster der Länge k+1, die F_i enthalten.

{ } D1, ..., D16

...{sun} {beach}

D1, D4, D5, D6, D2, D9, D13 D8, D10, D11, D15

D7, D14 D2, D9, D13 D8, D10, D11, D15

{sun,fun} {fun, beach} {sun,beach}

D1,D4,D6,D8 D10, D11, D13 ... D2, D8, D9, D10 D11, D15

{sun, fun, beach}

D8, D10, D11, D15

Anwendung für Tagging-Systeme

- Web 2.0: tagging
- Verbesserung von HFTS durch explizite Kriterien
  - Vollständigkeit
  - Überlappungsfreiheit

  als multikriterielle Optimierung
  Wurst/Kaspari in KDML 2007
- Verbessertes HFTS für die Strukturierung von Tagsets

Tagging

- Del.icio.us tagging von Web-Seiten
- Bibsonomy tagging von Web-Seiten und Literatur
- Last.FM tagging von Musik
- FLICKR tagging von Bildern
- YouTube tagging von Videos
- Yahoo! PODCASTS tagging von Podcasts
- TECHNORATI tagging von eigenen Blogs
- Upcoming tagging von Veranstaltungen

Folksonomy

- Eine Folksonomy ist ein Tupel (U,T,R,Y), wobei
- U Benutzer
- T tags
- R Ressourcen
- $Y \subseteq U \times T \times R$ und ein Tupel $(u,t,r) \in Y$ heißt, dass Benutzer u der Ressource r den tag t zugewiesen hat.

U	T	R
User1	tag1	Res2
User1	tag2	Res3
User2	tag3	Res3
User4	tag3	Res3

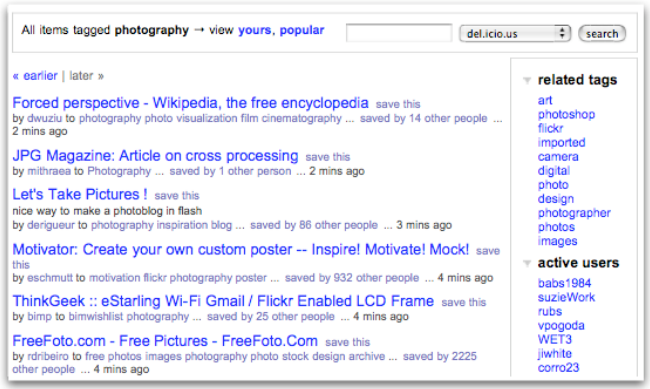
Häufigkeit berücksichtigen

.net advertising ajax apple architecture art article audio blog blogging blogs book books business comics community computer cooking cool css culture database design development diy download education english environment fashion fic finance firefox flash flickr fonts food free freeware fun funny furniture gallery game games google graphics green hardware health history home howto humor illustration imported inspiration internet java javascript jobs language library lifehacks linux mac magazine management maps marketing media microsoft mobile movies mp3 music network networking news online opensource osx phone photo photography photos photoshop php plugin politics portfolio productivity programming psychology python radio rails recipes reference research resource resources rss ruby rubyonrails science search security seo sga shop shopping slash social software tech technology tips tool tools toread travel tutorial tutorials tv twitter ubuntu video videos web web2.0 webdesign webdev wedding wiki windows wordpress work writing youtube

Reicht das?

- Kein Überblick
- Keine Gruppierung verwandter Begriffe
- Keine Navigationsmöglichkeit

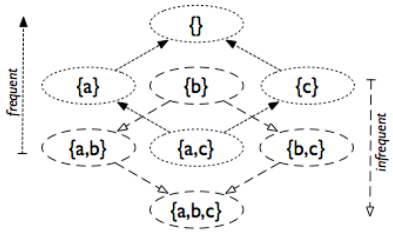
Tag “photography” Del.icio.us



Navigation

- Wie navigiert man in dieser Folksonomy?
  - Von “photography” kann man zu “art” navigieren (related links).
  - Man kann nicht von “photography” zu “photography UND art” navigieren!
- Wie bietet man hierarchische Navigationsstrukturen an?
- Natürlich durch HTFC!

Verband aller tagsets -- häufige

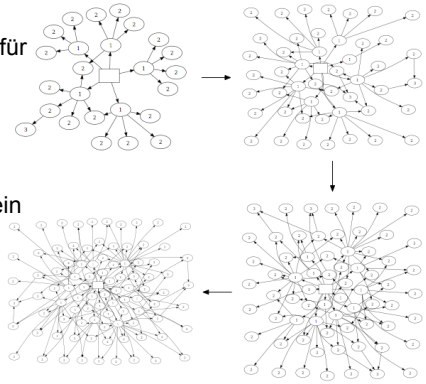


Problem

- Man erhält genau eine hierarchische Struktur häufiger tagsets.
- Verschiedene Benutzer möchten verschiedene Strukturen.
- Der Verband der häufigen tagsets wird nach verschiedenen Kriterien reduziert zum gewünschten clustering:
  - Vollständige Abdeckung von R (completeness)
  - Überschneidungsfreie cluster (overlap)
  - Ressourcen nicht nur im Wurzelknoten, sondern gut verteilt (coverage)
  - Niedrige Anzahl von Nachfolgern an jedem cluster (childcount)

Was ist schön?

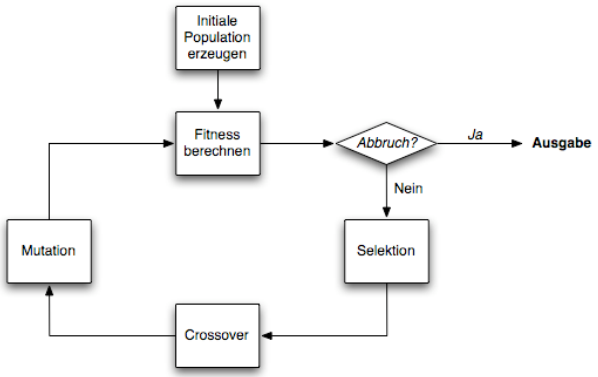
- Pareto-optimale Lösungen für die Optimierung nach
  - Anzahl Kinder vs.
  - Vollständigkeit
- Alle Lösungen werden für ein multi-kriterielles Problem in einem Lauf gewonnen.



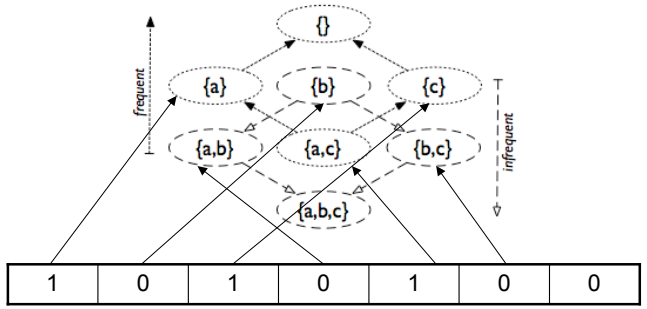
Multi-kriterielle Optimierung

- Gegensätzliche Gütekriterien
  - Nur empirisch feststellbar
  - Hier: Vollständigkeit vs. Anzahl Kinder und Verteiltheit der Ressourcen vs. Überschneidungsfreiheit
- Problem: Auswahl aus der Menge aller möglichen Teilmengen des FTS clusterings gemäß zweier gegensätzlicher Kriterien
- Lösung durch genetische Programmierung!

Genetische Programmierung



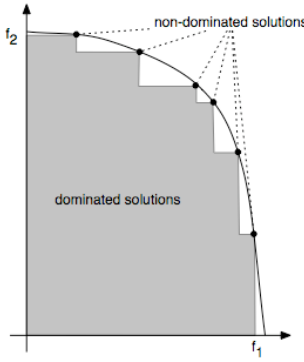
Individuen



Fitness Funktion

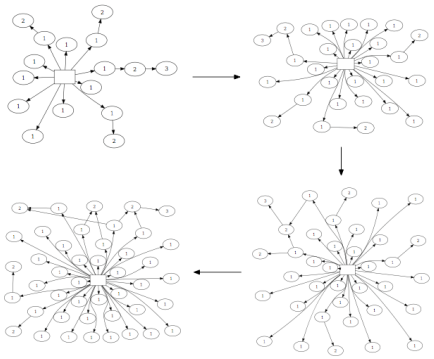
- Jedes Individuum x wird bewertet f: S --> R^n
- Vektor mit Gütemaßen für n Kriterien R^n
  Hier: Zwei reelle Werte, für jedes Kriterium eines.
- Pareto-Dominanz: Ein Vektor **u** dominiert einen Vektor **v**, wenn für alle i gilt $u_i \geq v_i$ und
  es gibt ein i, so dass $u_i > v_i$.
- Pareto-optimale Menge: Für ein multikriterielle Optimierungsproblem besteht die Pareto-optimale Menge aus allen nicht-dominierten Vektoren:
  $Q=\{x \in S \mid$ es gibt kein $y \in S$ mit $f(y) > f(x)\}$

Paretofront



Entlang der Paretofront

Overlap vs. coverage



Was wissen Sie jetzt?

- Man kann aus häufigen Mengen Clusterings gewinnen.
- Clusterings sind Teilmengen der Menge häufiger Mengen.
- Es gibt sehr viele Kriterien, nach denen man die Auswahl der Teilmengen steuern kann.
- Bei vielen Ansätzen sind die Kriterien fest in den Algorithmus eingebaut, z.B. durch Löschen schon einsortierter Ressourcen. Man erhält dann ein Clustering.
- Man kann aber auch gegensätzliche Kriterien mit GA optimieren und erhält verschiedene gleich gute Lösungen.