4 Exploiting Click Streams and Query Logs

Problem: Exploit Collective Human Input for Collaborative Web Search - Beyond Relevance Feedback and Beyond Google -

• href links are human endorsements → PageRank, etc.
• Opportunity: online analysis of human input & behavior may compensate deficiencies of search engine

Typical scenario for 3-keyword user query: a & b & c
→ top 10 results: user clicks on ranks 2, 5, 7
→ top 10 results: u
u
u
→ top 10 results: u
u
user asks friend for tips

query logs, bookmarks, etc. provide
• human assessments & endorsements
• correlations among words & concepts and among documents

Challenge: How can we use knowledge about the collective input of all users in a large community?

Problem 2: Exploit Collective Human Input for Automated Data(base Schema) Integration

• „semantic“ data integration is hoping for ontologies
• Opportunity: all existing DBs & apps already provide a large set of subjective mini-ontologies

Typical scenario for analyzing if A and B mean the same entity
→ compare their attributes, relationships, etc.
→ consider attribu
similar tables/d
→ consider instanc
similar tables/d
→ compare usage
in comparison to similar tables/docs of all known DBs

DB schemas, instances & data usage in apps provide
• human annotations
• correlations among tables, attr‘s, etc.

Challenge: How can we use knowledge about the collective designs of all DB apps in a large community?

4.2 Exploiting Click Streams

Simple idea: Modify HITS or Page-Rank algorithm by weighting edges with the relative frequency of users clicking on a link (as observed by DirectHit)

More sophisticated approach (Chen et al.:2002):
Consider link graph A and
link-visit matrix V (V_{ij}=1 if user i visits page j, 0 else)
Define
authority score vector: $a = \beta A^T h + (1-\beta)V^T u$
hub score vector: $h = \beta A a + (1-\beta)V^T u$
user importance vector: $u = (1-\beta)V(a+h)$
with a tunable parameter β (β=1: HITS, β=0: DirectHit)

claims to achieve higher precision than HITS, according to experimental results (with β=0.6) for some Webqueries such as „daily news“:
HITS top results: pricegrabber, gamespy, fileplanet, sportplanet, etc
Chen et al. method: news.com, bbc, cnn, google, lycos, etc.

Link Analysis based on Implicit Links (1)

Apply simple data mining to browsing sessions of many users, where each session i is a sequence (pi_1, pi_2, ...) of visited pages:
consider all pairs (pi_j, pi_{j+1}) of successively visited pages,
compute their total frequency f, and
selected those with f above some min-support threshold

Construct implicit-link graph with the selected page pairs as edges and their normalized total frequencies f as edge weights.

Apply edge-weighted Page-Rank for authority scoring, and linear combination of relevance and authority for overall scoring.

Link Analysis based on Implicit Links (2)

Experimental results (Xue et al.:2003):

performed on 4-month server-side (UC Berkeley) click-stream log with some „data cleaning“:
300 000 sessions of 60 000 users visiting 170 000 pages
with 200 000 explicit links

2-item frequent itemset mining yields
336 812 implicit links (incl. 22 122 explicit links)

Results for query „vision“:

	implicit PR	explicit PR	weighted HITS	DirectHit
1	vision group	some paper	Forsyth‘s book	workshop on vision
2	Forsysth‘s book	vision group	vision group	some student‘s resume
3	book 3rd edition	student resume	book 3rd edition	special course
4	workshop on vision	some talk slides	Leung‘s publ.	Forsyth‘s book
...				

not clear to me if any method is really better

4.3 Clustering Query Logs

Motivation:
- statistically identify FAQs (for intranets and portals), taking into account variations in query formulation
- capture correlation between queries and subsequent clicks

Model/Notation:
a user session is a pair (q, D+) with a query q and
D+ denoting the result docs on which the user clicked;
len(q) is the number of keywords in q

Similarity Measures between User Sessions

- tf*idf based similarity between query keywords only
- edit distance based similarity: sim(p,q) = 1 – ed(p,q) / max(len(p),len(q))

  Examples: Where does silk come from? Where does dew come from?
  How far away is the moon? How far away is the nearest star?
- similarity based on common clicks: $sim(p,q) = \frac{|D_p^+ \cap D_q^+|}{\max(|D_p^+|, |D_q^+|)}$

  Example: atomic bomb, Manhattan project, Nagasaki, Hiroshima, nuclear weapon
- similarity based on common clicks and document hierarchy:

$$sim(p,q) = \frac{1}{2}\left(\left(\sum_{d' \in D_p^+} \max\{s(d',d'') \mid d'' \in D_q^+\}\right) / |D_p^+| + \left(\sum_{d' \in D_q^+} \max\{s(d',d'') \mid d'' \in D_p^+\}\right) / |D_q^+|\right)$$

with $s(d',d'') = \frac{level(lca(d',d'')) - 1}{\max level - 1}$

p=law of thermodynamics
$D+_p$ = {/Science/Physics/Conservation Laws, ...}
q=Newton law
$D+_q$ = {/Science/Physics/Gravitation, ...}

- linear combinations of different similarity measures

Digression: K-Means Clustering Method

Idea:
- determine k prototype vectors, one for each cluster
- assign each data record to the most similar prototype vector and compute new prototype vector (e.g. by averaging over the vectors assigned to a prototype)
- iterate until clusters are sufficiently stable

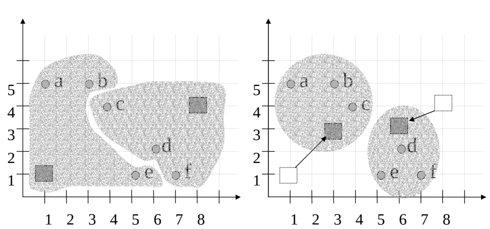
randomly choose k prototype vectors $\vec{c}_1, ..., \vec{c}_k$
while not yet sufficiently stable do
  for i:=1 to n do
    assign di to cluster cj for which $sim(\vec{d}_i, \vec{c}_j)$ is maximal
  od;
  for j:=1 to k do $\vec{c}_j := \frac{1}{|c_j|} \sum_{\vec{d} \in c_j} \vec{d}$ od;
od;

K-Means Clustering Method

- run-time is O(n) (assuming constant number of iterations)
- a suitable number of clusters, k, can be determined experimentally or based on the MDL principle or heuristic purity measures
- the initial prototype vectors could be chosen by using another – very efficient – clustering method
  (e.g. bottom-up clustering on random sample of the data records).
- for sim / dist any arbitrary metric can be used

Example for K-Means Clustering

K=2 data records prototype vectors



after 1st iteration after 2nd iteration

Incremental DBSCAN Clustering Method

based on M. Ester et al.: A density-based algorithm for discovering clusters in large spatial databases with noise, KDD Conference, 1996

simplified version of the algorithm:
for each data point d do {
  insert d into spatial index (e.g., R-tree);
  locate all points with distance to d < *max_dist*;
  if these points form a single cluster then add d to this cluster
  else {
    if there are at least *min_points* data points
      that do not yet belong to a cluster
      such that for all point pairs the distance < *max_dist*
    then construct a new cluster with these points };
};

average run-time is O(n * log n);
data points that are added later can be easily assigned to a cluster;
points that do not belong to any cluster are considered „noise“

Experimental Studies

performed on 20 000 queries against MS Encarta (an encyclopedia)

Observations:
- with sim threshold 1.0 the total number of clusters for the most popular 4500 queries (22%) was 400 for keyword sim and 200 for common-click sim
- combined keyword + common-click sim achieved best precision
- with sim threshold 0.6 the precision was above 90% (as intellectually assessed by „volunteers")

4.4 Exploiting Query Logs for Query Expansion

Given: user sessions of the form (q, D+), and let „d∈D+" denote the event that d is clicked on
We are interested in the correlation between words w in a query and w' in a clicked-on document:

$$P[w'|w] := P[w' \in d \;\; for \;\; some \;\; d \in D^+ | w \in q]$$
$$= \sum_{d \in D^+} \underbrace{P[w' \in d \,| d \in D^+]}_{\text{relative frequency of w' in d}} \cdot \underbrace{P[d \in D^+ | w \in q]}_{\text{relative frequency of d being clicked on when w appears in query}}$$

Estimate from query log:

Expand query by adding top m words w' in desc. order of $\prod_{w \in q} P[w'|w]$

Simple Alternative: Local Context Analysis based on Pseudo-Relevance Feedback

based on J. Xu and W.B. Croft: Improving the Effectiveness of Information Retrieval with Local Context Analysis, ACM TOIS Vol.18 No.1, 2000

Evaluate query q and extract from top k results:
select top m words or noun phrases
according to some tf*idf-style measure

Expand q by adding the selected words or noun phrases
(possibly with specific weights)

Experimental Evaluation

on MS Encarta corpus,
with 4 Mio. query log entries and 40 000 doc. subset

Considers short queries and long phrase queries, e.g.:

Michael Jordan	Michael Jordan in NBA matches
genome project	Why is the genome project so crucial for humans?
Manhattan project	What is the result of Manhattan project on Word War II?
Windows	What are the features of Windows that Microsoft brings us?

(Phrases are decomposed into N-grams that are in dictionary)

Avg. precision [%] at different recall values:

Short queries:

Recall	q alone	LC (n=100,m=30)	Query Log (m=40)
10%	40.67	45.00	62.33
20%	27.00	32.67	44.33
30%	20.89	26.44	36.78
100%	8.03	13.13	17.07

Long queries:

Recall	q alone	LC (n=100,m=30)	Query Log (m=40)
10%	46.67	41.67	57.67
20%	31.17	34.00	42.17
30%	25.67	27.11	34.89
100%	11.37	13.53	16.83

Digression: Association Rules

given:
a set of items I = {x1, ..., xm}
a set D ={t1, ..., tn} of item sets (transactions) ti = $\{xi_1, ..., xi_k\} \subseteq I$

wanted:
rules of the form X ⇒ Y with X ⊆ I and Y∈ I such that
- X is sufficiently often a subset of the item sets ti and
- when X ⊆ ti then most frequently Y∈ ti holds, too.

support (X ⇒ Y) = P[XY] = relative frequency of item sets that contain X and Y
confidence (X ⇒ Y) = P[Y|X] = relative frequency of item sets that contain Y provided they contain X

support is usually chosen in the range of 0.1 to 1 percent,
confidence (aka. strength) in the range of 90 percent or higher

Association Rules: Example

Market basket data („sales transactions"):

t1 = {Bread, Coffee, Wine}
t2 = {Coffee, Milk}
t3 = {Coffee, Jelly}
t4 = {Bread, Coffee, Milk}
t5 = {Bread, Jelly}
t6 = {Coffee, Jelly}
t7 = {Bread, Jelly}
t8 = {Bread, Coffee, Jelly, Wine}
t9 = {Bread, Coffee, Jelly}

support (Bread ⇒ Jelly) = 4/9 **confidence (Bread ⇒ Jelly) = 4/6**
support (Coffee ⇒ Milk) = 2/9 **confidence (Coffee ⇒ Milk) = 2/7**
support (Bread, Coffee ⇒ Jelly) = 2/9 **confidence (Bread, Coffee ⇒ Jelly) = 2/4**

Apriori Algorithm: Idea and Outline

Idea and outline:

- proceed in phases i=1, 2, ..., each making a single pass over D, and generate rules $X \Rightarrow Y$ with frequent item set X (sufficient support) and |X|=i in phase i;
- use phase i-1 results to limit work in phase i:
  antimonotonicity property (downward closedness):
  for i-item-set X to be frequent,
  each subset $X' \subseteq X$ with $|X'|=i-1$ must be frequent, too
- generate rules from frequent item sets;
- test confidence of rules in final pass over D

Worst-case time complexity is exponential in I and linear in D*I,
but usual behavior is linear in D
(detailed average-case analysis is very difficult)

Apriori Algorithm: Pseudocode

```
procedure apriori (D, min-support):
  L_1 = frequent 1-itemsets(D);
  for (k=2; L_{k-1} ≠ ∅; k++) {
     C_k = apriori-gen (L_{k-1}, min-support);
     for each t ∈ D { // linear scan of D
         C_t = subsets of t that are in C_k;
         for each candidate c ∈ C_t  {c.count++}; };
     L_k = {c ∈ C_k | c.count ≥ min-support}; };
  return L = ∪_k L_k; // returns all frequent item sets

procedure apriori-gen (L_{k-1}, min-support):
  C_k = ∅:
  for each itemset x_1 ∈ L_{k-1} {
    for each itemset x_2 ∈ L_{k-1} {
      if x_1 and x_2 have k-2 items in common and differ in 1 item // join {
         x = x_1 ∪ x_2;
         if there is a subset s ⊆ x with s ∉ L_{k-1} {disregard x;} // infreq. subset
         else add x to C_k; }; }; };
  return C_k
```


Algorithmic Extensions and Improvements

- hash-based counting (computed during very first pass):
  map k-itemset candidates (e.g. for k=2) into hash table and
  maintain one count per cell; drop candidates with low count early
- remove transactions that don‘t contain frequent k-itemset
  for phases k+1, ...
- partition transactions D:
  an itemset is frequent only if it is frequent in at least one partition
- exploit parallelism for scanning D
- randomized (approximative) algorithms:
  find all frequent itemsets with high probability (using hashing etc.)
- sampling on a randomly chosen subset of D

...

mostly concerned about reducing disk I/O cost
(for TByte databases of large wholesalers or phone companies)

Extensions and Generalizations of Assocation Rules

- quantified rules: consider quantitative attributes of item in transactions
  (e.g. wine between \$20 and \$50 $\Rightarrow$ cigars, or
  age between 30 and 50 $\Rightarrow$ married, etc.)
- constrained rules: consider constraints other than count thresholds,
  e.g. count itemsets only if average or variance of price exceeds ...
- generalized aggregation rules: rules referring to aggr. functions other
  than count, e.g., sum(X.price) $\Rightarrow$ avg(Y.age)
- multilevel association rules: considering item classes
  (e.g. chips, peanuts, bretzels, etc. belonging to class snacks)
- sequential patterns
  (e.g. an itemset is a customer who purchases books in some order,
  or a tourist visiting cities and places)
- from strong rules to interesting rules:
  consider also lift (aka. interest) of rule $X \Rightarrow Y$: P[XY] / P[X]P[Y]
- correlation rules
- causal rules

Correlation Rules

example for strong, but misleading association rule:
tea $\Rightarrow$ coffee with confidence 80% and support 20%
but support of coffee alone is 90%, and of tea alone it is 25%
$\rightarrow$ tea and coffee have negative correlation !

consider contingency table (assume n=100 transactions):

	T	¬T	
C	20	70	90
¬C	5	5	10
	25	75	

$\rightarrow$ {T, C} is a frequent and correlated item set

$$\chi^2(C,T) = \sum_{X \in \{C,\overline{C}\}} \sum_{Y \in \{T,\overline{T}\}} (\frac{(freq(X \wedge Y) - freq(X)\,freq(Y)/n)^2}{freq(X)\,freq(Y)/n}$$

correlation rules are monotone (upward closed):
if the set X is correlated then every superset $X' \supseteq X$ is correlated, too.

Correlated Item Set Algorithm

```
procedure corrset (D, min-support, support-fraction, significance-level):
  for each x ∈ I compute count O(x);
  initialize candidates := ∅; significant := ∅;
  for each item pair x, y ∈ I with O(x) > min-support and O(y) > min-support {
      add (x,y) to candidates; };
  while (candidates ≠ ∅) {
      notsignificant := ∅;
      for each itemset X ∈candidates {
          construct contingency table T;
          if (percentage of cells in T with count > min-support
          is at least support-fraction) { // otherwise too few data for chi-square
              if (chi-square value for T ≥ significance-level)
              {add X to significant} else {add X to notsignificant};
          }; //if
      }; //for
      candidates := itemsets with cardinality k such that
                    every subset of cardinality k-1 is in notsignificant;
                    // only interested in correlated itemsets of min. cardinality
  }; //while
  return significant
```

Frequent Itemset and Correlated Itemsets Applied to Query Logs

Infer from user sessions of the form (q, D+) where q is a set of words association rules of the form:

w_1 and w_2 => w_3

Infer from user sessions of the form (q, D+)
where q is a set of „signed“ (positive or negative) words
correlation rules of the form:

$sign_1$ w_1 and $sign_2$ w_2 => $sign_3$ w_3

where $sign_i$ is either + or – and indicates positive or negative correlation

Expand new query with word set W by right-hand sides r of association rules L => r for which $L \subseteq W$

Literature

- Zheng Chen, Li Tao, Jidong Wang, Liu Wenyin, Wei-Ying Ma: A Unified Framework for Web Link Analysis, WISE Conf., 2002
- Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Wei-Ying Ma, Hong-Jiang Zhang, Chao-Jun Lu: Implicit Link Analysis for Small Web Search, SIGIR Conf., 2003
- Ji-Rong Wen, Jian-Yun Nie, Hong-Jiang Zhang: Query Clustering Using User Logs, ACM TOIS Vol.20 No.1, 2002
- Hang Cui, Ji-Rong Wen, Jian-Yun Nie, Wei-Ying Ma: Query Expansion by Mining User Logs, IEEE Transactions on Knowledge and Data Engineering Vol.15 No.4, 2003