+

Databases and Information Retrieval Integration

TIETS42

Rank Aggregation

Kostas Stefanidis
kostas.stefanidis@uta.fi

Autumn 2016

http://www.uta.fi/sis/tie/dbir/index.html

http://people.uta.fi/~kostas.stefanidis/dbir16/dbir16-main.html

Motivation

- Huge amounts of available information
- Users search for interesting information without knowing the content and structure of the sources
- Users consume typically the “best” (first) matches

Rank data!

+

Databases and Information Retrieval Integration

TIETS42

Rank Aggregation

Kostas Stefanidis
kostas.stefanidis@uta.fi

Autumn 2016

http://www.uta.fi/sis/tie/dbir/index.html

http://people.uta.fi/~kostas.stefanidis/dbir16/dbir16-main.html

Rank Aggregation

Given: A set of rankings $R_1, R_2, \ldots, R_m$ of a set of objects $X_1, X_2, \ldots, X_n$

Produce: A single ranking R that is in agreement with the existing rankings

Examples

- Voting: Rankings $R_1, R_2, \ldots, R_m$ are the voters, the objects $X_1, X_2, \ldots, X_n$ are the candidates
- Combining multiple scoring functions: Rankings $R_1, R_2, \ldots, R_m$ are the scoring functions, the objects $X_1, X_2, \ldots, X_n$ are data items
  - Combine scores for multimedia items
    - Color, shape, texture
  - Combine scores for database tuples
    - Find the best hotel according to price and location

Variants of the Problem

- Combining scores
  - We know the scores assigned to objects by each ranking, and we want to compute a single score
- Combining rankings
  - The scores are not known, only the ordering is known

Combining Scores

- Each object X_i has m scores $(r_{i1}, r_{i2}, \ldots, r_{im})$
- The score of object X_i is computed using an aggregate scoring function $f(r_{i1}, r_{i2}, \ldots, r_{im})$

	R_1	R_2	R_3
X_1	1	0.3	0.2
X_2	0.8	0.8	0
X_3	0.5	0.7	0.6
X_4	0.3	0.2	0.8
X_5	0.1	0.1	0.1

Combining Scores

- Each object X_i has m scores $(r_{i1}, r_{i2}, \ldots, r_{im})$
- The score of object X_i is computed using an **aggregate scoring function** $f(r_{i1}, r_{i2}, \ldots, r_{im})$
  - $f(r_{i1}, r_{i2}, \ldots, r_{im}) = \min\{r_{i1}, r_{i2}, \ldots, r_{im}\}$

	R_1	R_2	R_3	R
X_1	1	0.3	0.2	0.2
X_2	0.8	0.8	0	0
X_3	0.5	0.7	0.6	0.5
X_4	0.3	0.2	0.8	0.2
X_5	0.1	0.1	0.1	0.1

Combining Scores

- Each object X_i has m scores $(r_{i1}, r_{i2}, \ldots, r_{im})$
- The score of object X_i is computed using an **aggregate scoring function** $f(r_{i1}, r_{i2}, \ldots, r_{im})$
  - $f(r_{i1}, r_{i2}, \ldots, r_{im}) = \max\{r_{i1}, r_{i2}, \ldots, r_{im}\}$

	R_1	R_2	R_3	R
X_1	1	0.3	0.2	1
X_2	0.8	0.8	0	0.8
X_3	0.5	0.7	0.6	0.7
X_4	0.3	0.2	0.8	0.8
X_5	0.1	0.1	0.1	0.1

Combining Scores

- Each object X_i has m scores $(r_{i1}, r_{i2}, \ldots, r_{im})$
- The score of object X_i is computed using an **aggregate scoring function** $f(r_{i1}, r_{i2}, \ldots, r_{im})$
  - $f(r_{i1}, r_{i2}, \ldots, r_{im}) = r_{i1} + r_{i2} + \ldots + r_{im}$

	R_1	R_2	R_3	R
X_1	1	0.3	0.2	1.5
X_2	0.8	0.8	0	1.6
X_3	0.5	0.7	0.6	1.8
X_4	0.3	0.2	0.8	1.3
X_5	0.1	0.1	0.1	0.3

Top-k

Given a set of n objects and m scoring lists sorted in decreasing order, *find the top-k objects according to a scoring function f*

top-k: a set T of k objects such that $f(r_{j1},\ldots,r_{jm}) \leq f(r_{i1},\ldots,r_{im})$ for every object X_i in T and every object X_j not in T

Assumption: The function f is monotone

- $f(r_1,\ldots,r_m) \leq f(r_1',\ldots,r_m')$ if $r_i \leq r_i'$ for all i

Objective: Compute top-k with the minimum cost

Cost function

We want to minimize the number of accesses to the scoring lists

- Sorted accesses: sequentially access the objects in the order in which they appear in a list
  - cost C_s
- Random accesses: obtain the cost value for a specific object in a list
  - cost C_r
- If s sorted accesses and r random accesses minimize $s\ C_s + r\ C_r$

Example

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

- Compute top-2 for the sum aggregate function

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are **k** objects that have been seen in **all** lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are **k** objects that have been seen in **all** lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are **k** objects that have been seen in **all** lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are **k** objects that have been seen in **all** lists

R_1		R_2		R_3	
X_1	1	X_2	0.8	X_4	0.8
X_2	0.8	X_3	0.7	X_3	0.6
X_3	0.5	X_1	0.3	X_1	0.2
X_4	0.3	X_4	0.2	X_5	0.1
X_5	0.1	X_5	0.1	X_2	0

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are **k** objects that have been seen in **all** lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin's Algorithm

2. Perform random accesses to obtain the scores of all seen objects

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

+ Fagin's Algorithm

3. Compute score for all objects and find the top-k

R1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

R	
X_3	1.8
X_2	1.6
X_1	1.5
X_4	1.3

Fagin's Algorithm

- X_5 cannot be in the top-2 because of the monotonicity property
  - $f(X_5) \leq f(X_1) \leq f(X_3)$

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

R	
X_3	1.8
X_2	1.6
X_1	1.5
X_4	1.3

+ Threshold algorithm

1. Access the elements sequentially

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Threshold algorithm

1. At each sequential access
   a. Set the threshold **t** to be the aggregate of the scores seen in this access

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 2.6$

Threshold algorithm

1. At each sequential access
   b. Do random accesses and compute the score of the objects seen

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 2.6$

X_1	1.5
X_2	1.6
X_4	1.3

Threshold algorithm

1. At each sequential access
   c. Maintain a list of top-k objects seen so far

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 2.6$

X_2	1.6
X_1	1.5

Threshold algorithm

1. At each sequential access
   d. When the scores of the top-k are greater or equal to the threshold, stop

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 2.1$

X_3	1.8
X_2	1.6

Threshold algorithm

1. At each sequential access
   d. When the scores of the top-k are greater or equal to the threshold, stop

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 1.0$

X_3	1.8
X_2	1.6

Threshold algorithm

2. Return the top-k seen so far

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

t = 1.0

X_3	1.8
X_2	1.6

Threshold algorithm

- From the monotonicity property for any object not seen, the score of the object is less than the threshold
  - $f(X_5) \leq t \leq f(X_2)$

Combining Rankings

In many cases, the scores are not known, or we do not know how they were obtained

- one search engine returns score 10, the other 100. What does this mean?

Work with the rankings

The Problem

Input: A set of rankings $R_1,R_2,\ldots,R_m$ of the objects $X_1,X_2,\ldots,X_n$

- Each ranking R_i is a total ordering of the objects
  - For every pair X_i,X_j either X_i is ranked above X_j or X_j is ranked above X_i

Output: A total ordering R that aggregates rankings $R_1,R_2,\ldots,R_m$

Voting theory

A voting system is a rank aggregation mechanism

Long history and literature

- criteria and axioms for good voting systems

What is a good voting system?

- The Condorcet criterion
  - If object A defeats every other object in a pair-wise majority vote, then A should be ranked first
- Extended Condorcet criterion
  - If the objects in a set X defeat in pair-wise comparisons the objects in the set Y then the objects in X should be ranked above those in Y
- Not all voting systems satisfy the Condorcet criterion!

Pair-wise majority comparisons

Unfortunately the Condorcet winner does not always exist

- Irrational behavior of groups

	V_1	V_2	V_3
1	A	B	C
2	B	C	A
3	C	A	B

A > B B > C C > A

Pair-wise majority comparisons

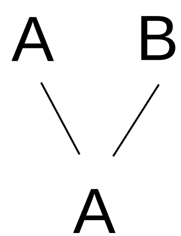
- Resolve cycles by imposing an agenda

	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D

+ Pair-wise majority comparisons

- Resolve cycles by imposing an agenda

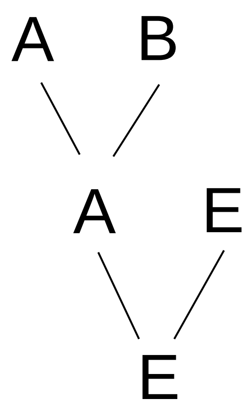
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



Pair-wise majority comparisons

- Resolve cycles by imposing an agenda

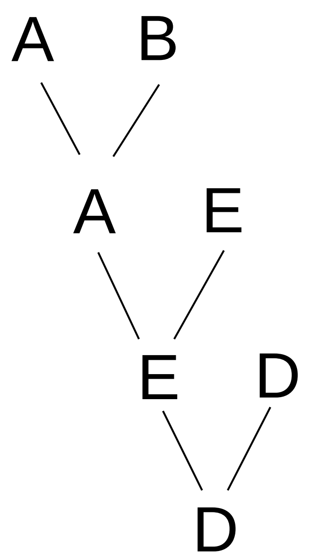
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



Pair-wise majority comparisons

- Resolve cycles by imposing an agenda

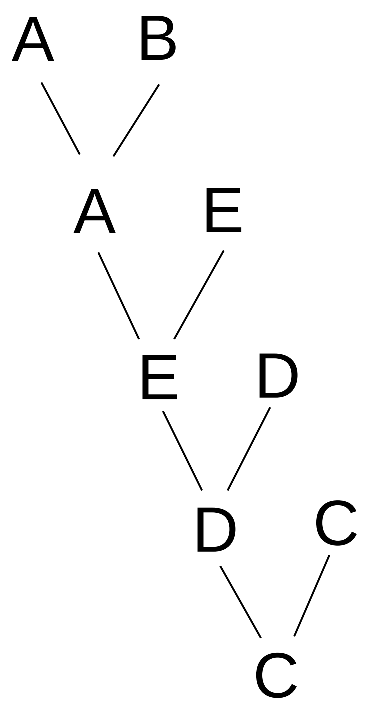
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



Pair-wise majority comparisons

- Resolve cycles by imposing an agenda

	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D

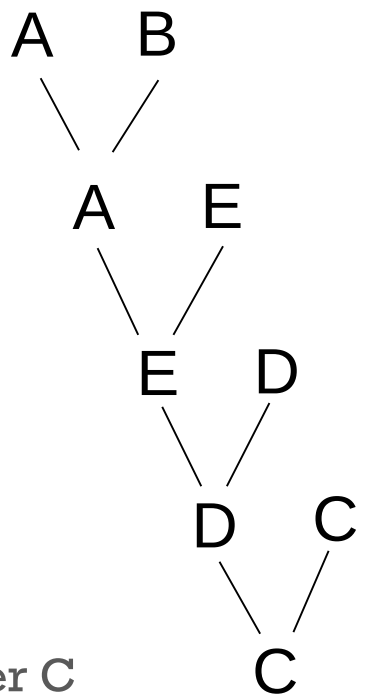


- C is the winner

+ Pair-wise majority comparisons

- Resolve cycles by imposing an agenda

	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



- But everybody prefers A or B over C

Pair-wise majority comparisons

There exists another ordering that everybody prefers!

Plurality vote

- Elect first whoever has more 1st position votes

voters	10	8	7
1	A	C	B
2	B	A	C
3	C	B	A

Plurality with runoff

- If no-one gets more than 50% of the 1st position votes, take the majority winner of the first two

voters	10	8	7	2
1	A	C	B	B
2	B	A	C	A
3	C	B	A	C

first round: A 10, B 9, C 8
second round: A 18, B 9
winner: A

remove C from the 1st position of column2

Plurality with runoff

- If no-one gets more than 50% of the 1st position votes, take the majority winner of the first two

voters	10	8	7	2
1	A	C	B	A
2	B	A	C	B
3	C	B	A	C

change the order of A and B in the last column

first round: A 12, B 7, C 8
second round: A 12, C 15
winner: C!

Borda Count

- For each ranking, assign to object X, number of points equal to the number of objects it defeats
  - first position gets $n-1$ points, second $n-2$, …, last 0 points
- The total weight of X is the number of points it accumulates from all rankings

Borda Count

voters	3	2	2
1 (3p)	A	B	C
2 (2p)	B	C	D
3 (1p)	C	D	A
4 (0p)	D	A	B

A: 3*3 + 2*0 + 2*1 = 11p
B: 3*2 + 2*3 + 2*0 = 12p
C: 3*1 + 2*2 + 2*3 = 13p
D: 3*0 + 2*1 + 2*2 = 6p

BC
C
B
A
D

Borda Count

- Assume that D is removed from the vote

voters	3	2	2
1 (2p)	A	B	C
2 (1p)	B	C	A
3 (0p)	C	A	B

A: 3*2 + 2*0 + 2*1 = 8p
B: 3*1 + 2*2 + 2*0 = 7p
C: 3*0 + 2*1 + 2*2 = 6p

BC
A
B
C

- Changing the position of D changes the order of the other elements!

Borda Count

The Borda Count of an object X is the aggregate number of pair-wise comparisons that the object X wins

- Follows from the fact that in one ranking X wins all the pair-wise comparisons with objects that are under X in the ranking

Kemeny Optimal Aggregation

Kemeny distance $K(R_1,R_2)$: The number of pairs of nodes that are ranked in a different order (Kendall-tau)

- number of swaps required to transform one ranking into another (or number of pairs of tuples with different order between the rankings)

Kemeny optimal aggregation minimizes

$$K(R, R_1, \ldots, R_m) = \sum_{i=1}^{m} K(R, R_i)$$

Spearman's footrule distance

Spearman's footrule distance: The difference between the ranks R(i) and R'(i) assigned to object i

$$F(R, R') = \sum_{i=1}^{n} \left| R(i) - R'(i) \right|$$

Relation between Spearman's footrule and Kemeny distance

$$K(R, R') \leq F(R, R') \leq 2K(R, R')$$

Spearman's footrule aggregation

Find the ranking R, that minimizes

$$F(R, R_1, \ldots, R_m) = \sum_{i=1}^{m} F(R, R_i)$$

+ Example

S = {A,B,C,D,E}

σ , τ : two full list

	σ	τ
1	A	C
2	C	A
3	E	B
4	D	D
5	B	E

- Spearman's Footrule Distance
  - F(σ , τ) = 1 + 2 + 1 + 0 + 2 = 6
- Kendall tau distance
  - K(σ , τ) = |{(A,C), (B,D), (B,E), (D,E)}| = 4

+ Homework (1/3)

Compare the rankings in cases 1 and 2. Use both the Kendall tau distance and the Spearman footrule distance

Case 1

	σ	τ
1	A	D
2	C	A
3	D	C
4	B	E
5	E	B

Case 2

	σ	τ
1	X	C
2	C	A
3	E	B
4	D	D
5	K	E

+ Homework (2/3)

Find the top-2 objects, using (i) the Fagin's and (ii) the Threshold algorithms. Show all intermediate steps. Use the *sum* aggregation function.

R1	
X1	1
X2	0.9
X3	0.6
X4	0.5
X5	0.4
X6	0.2
X7	0

R2	
X2	0.9
X1	0.8
X4	0.6
X3	0.4
X5	0.3
X7	0.2
X6	0.1

R3	
X3	1
X1	0.9
X5	0.8
X2	0.7
X6	0.6
X7	0.5
X4	0.4

R4	
X2	0.9
X3	0.7
X1	0.6
X5	0.5
X6	0.4
X7	0.2
X4	0

Homework (2/3)

Send your solutions at kostas.stefanidis@uta.fi

Before NOVEMBER 4, 2016.