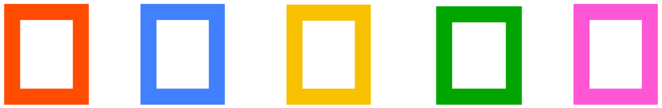
More on Rankings

- Comparing results of Link Analysis Ranking algorithms

- Comparing and aggregating rankings

Comparing LAR vectors



$$w_1 = [\ 1 \quad 0.8 \quad 0.5 \quad 0.3 \quad 0\ ]$$

$$w_2 = [\ 0.9 \quad 1 \quad 0.7 \quad 0.6 \quad 0.8\ ]$$

- How close are the LAR vectors w_1, w_2?

Distance between LAR vectors

- Geometric distance: how close are the numerical weights of vectors w_1, w_2?

$$d_1(w_1, w_2) = \sum \left| w_1[i] - w_2[i] \right|$$

Distance between LAR vectors

- Geometric distance: how close are the numerical weights of vectors w_1, w_2?

$$d_1(w_1, w_2) = \sum |w_1[i] - w_2[i]|$$

$$w_1 = [\ 1.0 \quad 0.8 \quad 0.5 \quad 0.3 \quad 0.0\ ]$$

$$w_2 = [\ 0.9 \quad 1.0 \quad 0.7 \quad 0.6 \quad 0.8\ ]$$

Distance between LAR vectors

- Geometric distance: how close are the numerical weights of vectors w_1, w_2?

$$d_1(w_1, w_2) = \sum \left| w_1[i] - w_2[i] \right|$$

$$w_1 = [\ 1.0 \quad 0.8 \quad 0.5 \quad 0.3 \quad 0.0\ ]$$

$$w_2 = [\ 0.9 \quad 1.0 \quad 0.7 \quad 0.6 \quad 0.8\ ]$$

$$d_1(w_1,w_2) = \quad 0.1+0.2+0.2+0.3+0.8 = 1.6$$

Distance between LAR vectors

- Rank distance: how close are the ordinal rankings induced by the vectors w_1, w_2?
  - Kendal's τ distance

$$d_r(w_1, w_2) = \frac{\text{pairs ranked in a different order}}{\text{total number of distinct pairs}}$$

Outline

- Rank Aggregation
  - Computing aggregate scores
  - Computing aggregate rankings – voting

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.

Examples

- 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 the PageRank scores with term-weighting scores
    - Combine scores for multimedia items
      - color, shape, texture
    - Combine scores for database tuples
      - find the best hotel according to price and location

Examples

- Combining multiple sources
  - rankings $R_1, R_2, \ldots, R_m$ are the sources, the objects $X_1, X_2, \ldots, X_n$ are data items.
    - meta-search engines for the Web
    - distributed databases
    - P2P sources

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 ordinal rankings
  - the scores are not known, only the ordering is known
  - the scores are known but we do not know how, or do not want to combine them
    - e.g. price and star rating

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

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	
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

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

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

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

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

Fagin’s Algorithm

- The algorithm is cost optimal under some probabilistic assumptions for a restricted class of aggregate functions

Threshold algorithm

1. Access the elements sequentially

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
  - $f(X_5) \leq t \leq f(X_2)$
- The algorithm is instance cost-optimal
  - within a constant factor of the best algorithm on any database

Combining rankings

- In many cases the scores are not known
  - e.g. meta-search engines - scores are proprietary information
- … or we do not know how they were obtained
  - one search engine returns score 10, the other 100. What does this mean?
- … or the scores are incompatible
  - apples and oranges: does it make sense to combine price with distance?

- In this cases we can only 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 pairwise majority vote, then A should be ranked first
- Extended Condorcet criterion
  - if the objects in a set X defeat in pairwise 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!

Pairwise 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

Pairwise majority comparisons

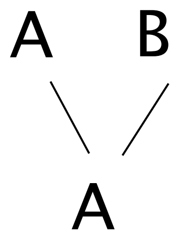
- Resolve cycles by imposing an agenda

	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

Pairwise majority comparisons

- Resolve cycles by imposing an agenda

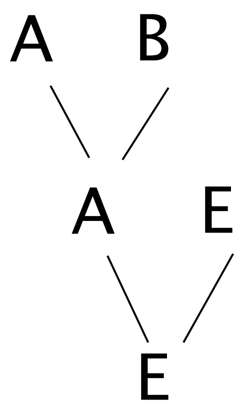
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



Pairwise majority comparisons

- Resolve cycles by imposing an agenda

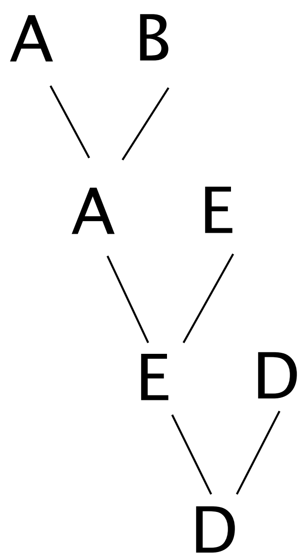
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



Pairwise majority comparisons

- Resolve cycles by imposing an agenda

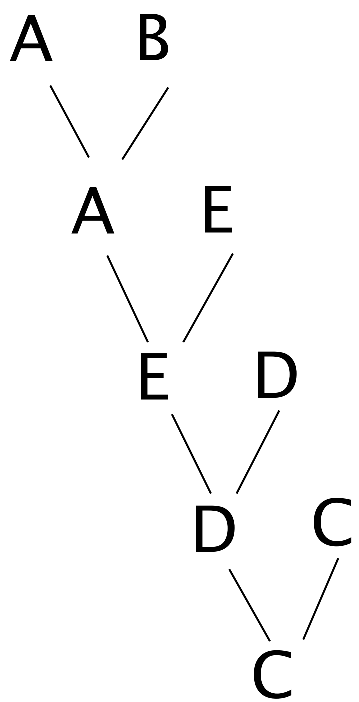
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



Pairwise 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

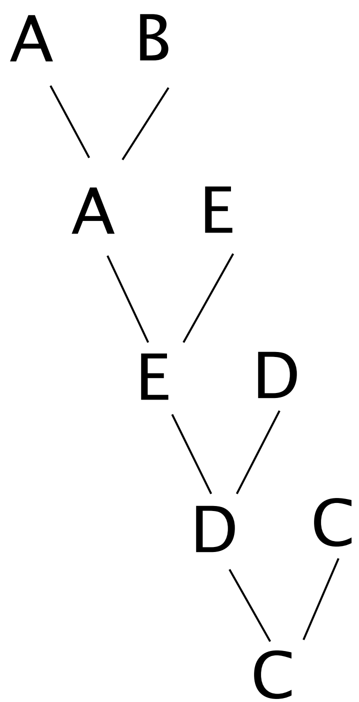


- C is the winner

Pairwise 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

Pairwise majority comparisons

- The voting system is not Pareto optimal
  - there exists another ordering that everybody prefers

- Also, it is sensitive to the order of voting

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

- Does not find a Condorcet winner (C in this case)

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

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!

Positive Association axiom

- Plurality with runoff violates the positive association axiom

- Positive association axiom: positive changes in preferences for an object should not cause the ranking of the object to decrease

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

- Does not always produce Condorcet winner

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

BC
B
A
C

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

Independence of Irrelevant Alternatives

- The relative ranking of X and Y should not depend on a third object Z
  - heavily debated axiom

Borda Count

- The Borda Count of an an object X is the aggregate number of pairwise comparisons that the object X wins
  - follows from the fact that in one ranking X wins all the pairwise comparisons with objects that are under X in the ranking

Voting Theory

- Is there a voting system that does not suffer from the previous shortcomings?

Arrow’s Impossibility Theorem

- No voting system satisfies the following axioms
  - Universality
    - all inputs are possible
  - Completeness and Transitivity
    - for each input we produce an answer and it is meaningful
  - Positive Assosiation
    - Promotion of a certain option cannot lead to a worse ranking of this option.
  - Independence of Irrelevant Alternatives
    - Changes in individuals' rankings of irrelevant alternatives (ones outside a certain subset) should have no impact on the societal ranking of the subset.
  - Non-imposition
    - Every possible societal preference order should be achievable by some set of individual preference orders
  - Non-dictatoriship
- **KENNETH J. ARROW** Social Choice and Individual Values (1951). Won Nobel Prize in 1972

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 bubble-sort swaps required to transform one ranking into another
- Kemeny optimal aggregation minimizes

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

- Kemeny optimal aggregation satisfies the Condorcet criterion and the extended Condorcet criterion
  - maximum likelihood interpretation: produces the ranking that is most likely to have generated the observed rankings
- …but it is NP-hard to compute
  - easy 2-approximation by obtaining the best of the input rankings, but it is not "interesting"

Rankings as pairwise comparisons

- If element u is **before** element v, then u is preferred to v

- From input rankings output **majority tournaments** $G = (U,A)$:
  - for u,v in U, if the majority of the rankings prefer u to v, then add (u,v) to A

The KwikSort algorithm

- KwikSort(G=(U,A))
  - if U is empty return empty list
  - U1 = U2 = empty set
  - pick random pivot u from U
  - For all v in U\{u}
    - if (v,u) is in A then add v to U1
    - else add v to U2
  - G1 = (U1,A1)
  - G2 = (U2,A2)
  - Return KwikSort(G1),u,KwikSort(G2)

Properties of the KwikSort algorithm

- KwikSort algorithm is a 3-approximation algorithm to the Kemeny aggregation problem

Locally Kemeny optimal aggregation

- A ranking R is locally Kemeny optimal if there is no bubble-sort swap of two consecutively placed objects that produces a ranking R' such that
- $K(R',R_1,\ldots,R_m) \leq K(R,R_1,\ldots,R_m)$

- Locally Kemeny optimal is not necessarily Kemeny optimal
- Definitions apply for the case of partial lists also

Locally Kemeny optimal aggregation

- Locally Kemeny optimal aggregation can be computed in polynomial time
  - At the i-th iteration insert the i-th element x in the bottom of the list, and bubble it up until there is an element y such that the majority places y over x

- Locally Kemeny optimal aggregation satisfies the Condorcet and extended Condorcet criterion

Rank Aggregation algorithm [DKNS01]

- Start with an aggregated ranking and make it into a locally Kemeny optimal aggregation
- How do we select the initial aggregation?
  - Use another aggregation method
  - Create a Markov Chain where you move from an object X, to another object Y that is ranked higher by the majority

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

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

- The optimal Spearman's footrule aggregation can be computed in polynomial time
  - It also gives a 2-approximation to the Kemeny optimal aggregation

- If the median ranks of the objects are unique then this ordering is optimal

Example

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

A: (1 , 2 , 3)
B: (1 , 1 , 2)
C: (2 , 3 , 4)
D: (3 , 4 , 4)

Example

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

A: $(1 , \boxed{2} , 3)$
B: $(1 , \boxed{1} , 2)$
C: $(2 , \boxed{3} , 4)$
D: $(3 , \boxed{4} , 4)$

Example

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	A
3	C
4	D

A: (1 , $\boxed{2}$, 3)
B: (1 , $\boxed{1}$, 2)
C: (2 , $\boxed{3}$, 4)
D: (3 , $\boxed{4}$, 4)

The MedRank algorithm

- Access the rankings sequentially

R_1		R_2		R_3		R	
1	A	1	B	1	B	1	
2	B	2	A	2	C	2	
3	C	3	D	3	A	3	
4	D	4	C	4	D	4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	
3	
4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	A
3	
4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	A
3	C
4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	A
3	C
4	D

The Spearman's rank correlation

- Spearman's rank correlation

$$S(R, R') = \sum_{i=1}^{n} \left(R(i) - R'(i)\right)^2$$

- Computing the optimal rank aggregation with respect to Spearman's rank correlation is the same as computing Borda Count
  - Computable in polynomial time

Extensions and Applications

- Rank distance measures between partial orderings and top-k lists
- Similarity search
- Ranked Join Indices
- Analysis of Link Analysis Ranking algorithms
- Connections with machine learning

References

- A. Borodin, G. Roberts, J. Rosenthal, P. Tsaparas, Link Analysis Ranking: Algorithms, Theory and Experiments, ACM Transactions on Internet Technologies (TOIT), 5(1), 2005
- Ron Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, Erik Vee, Comparing and aggregating rankings with ties , PODS 2004
- M. Tennenholtz, and Alon Altman, "On the Axiomatic Foundations of Ranking Systems", Proceedings of IJCAI, 2005
- Ron Fagin, Amnon Lotem, Moni Naor. Optimal aggregation algorithms for middleware, J. Computer and System Sciences 66 (2003), pp. 614-656. Extended abstract appeared in Proc. 2001 ACM Symposium on Principles of Database Systems (PODS '01), pp. 102-113.
- Alex Tabbarok Lecture Notes
- Ron Fagin, Ravi Kumar, D. Sivakumar Efficient similarity search and classification via rank aggregation, Proc. 2003 ACM SIGMOD Conference (SIGMOD '03), pp. 301-312.
- Cynthia Dwork, Ravi Kumar, Moni Naor, D. Sivakumar. Rank Aggregation Methods for the Web. 10th International World Wide Web Conference, May 2001.
- C. Dwork, R. Kumar, M. Naor, D. Sivakumar, "Rank Aggregation Revisited," WWW10; selected as Web Search Area highlight, 2001.