Top-*k* Processing for Search and Information Discovery in Social Applications

Lecture 3: Group Recommendation

Sihem Amer-Yahia

مؤسسة قطر
Qatar Foundation

Julia Stoyanovich

Penn
University of Pennsylvania

Social top-*k* @ *Joint RuSSIR/EDBT Summer School 2011*

Summary of last lectures

- **Semantics of top-*k* queries**
  - Items have score that are made up of components
  - Components are aggregated using monotone aggregation
- **Fundamental algorithms**
  - Use the inverted list indexing structure
  - Have an access strategy and a stopping condition
  - TA – instance-optimal over the class of *reasonable* algorithms
  - NRA – useful when random access is expensive or impossible
- **Network-aware search**
  - Ubiquitous on the Social Web
  - Careful modeling of inverted lists enables top-*k* applicability
  - Space/time tradeoff exploration for scalable network-aware search (Cluster-Seekers and Cluster-Taggers)

Quote of the day

I don't want to be a member of a club

that would have me as a member.

~Groucho Marx via Woody Allen

Group recommendation

- **How do you decide where to go to dinner with friends?**
  - email/text/phone
  - not optimal for reaching consensus
- **What if there was a system that knew each user's preferred list?**
- **What is the best way to compute a group's preferred list?**
- **How to *efficiently* do that?**

Group recommendation by example

- **Task: recommend a movie to group G ={u1, u2 ,u3}**
  - predictedRating(u1,"God Father") = 5
  - predictedRating(u2, "God Father") = 1
  - predictedRating(u3, "God Father") = 1

  - predictedRating(u1, "Roman Holiday") = 3
  - predictedRating(u2, "Roman Holiday") = 3
  - predictedRating(u3, "Roman Holiday") = 1

- ***Average Rating*** **and** ***Least Misery*** **fail to distinguish between "God Father" and "Roman Holiday"**

Outline

- ✓ Intro
- Problem definition
- Top-*k* applicability
- Performance optimizations
- Experiments

Introducing group consensus

Consensus function *combines* ***relevance*** *(average or least misery) and* ***disagreement*** *(average pair-wise or variance) in the score of a group recommendation*

$\mathcal{F}(\mathcal{G}, i) = w_1 \times \texttt{rel}(\mathcal{G}, i) + w_2 \times (1 - \texttt{dis}(\mathcal{G}, i))$, *where* $w_1 + w_2 = 1.0$ *and each specifies the relative importance of relevance and disagreement in the overall recommendation score.*

- ***Different from computing user affinities to find implicit networks*** ***[see slide 13 from Lecture 2]***
- ***Consensus is computed per item and for groups formed in an ad-hoc fashion***

Problem definition

Given an ad-hoc user group G and a consensus function F, find the k best items according to F, such that each item is new to all users in G.

Outline

- ✓ Intro
- ✓ Problem definition
- **Top-k applicability**
  - Enforcing monotonicity
  - Performance bottleneck
- **Performance optimizations**
- **Experiments**

Top-*k* applicability

- **Average and Least Misery are monotone**
- **Input: 3 relevance lists (IL_{u1} ,IL_{u2} ,IL_{u3})**
  - sorted on decreasing value of user's predicted rating
  - apply Fagin top-k algorithm (e.g., NRA)

IL_{U1}	IL_{U2}	IL_{U3}
i1,4	i2,4	I4,8
i3,4	i4,4	I1,5
i4,4	i1,2	i3,3
i2,2	i3,2	i2,3

Relevance-Only (RO)

- **Input: 3 relevance lists (IL_{u1} ,IL_{u2} ,IL_{u3})**
  - problem: no pruning
    - disagreement component of scoring function is not monotone! **[see slide 7 from Lecture 1]**
  - intuition: pruning only when disagreement "correlates" with score

IL_{U1}	IL_{U2}	IL_{U3}
i1,4	i2,4	I4,8
i3,4	i4,4	I1,5
i4,4	i1,2	i3,3
i2,2	i3,2	i2,3

Relevance Only (RO) algorithm

IL_{u1}	IL_{u2}	IL_{u3}
i1,4	i2,4	i3,3
i3,4	i4,4	i1,3
i4,4,	i1,2	i4,3
i2,2	i3,2	i2,3

Top-k Buffer
i1,1.33

Threshold is : 1.93

Relevance Only (RO) algorithm

IL_{u1}	IL_{u2}	IL_{u3}
i1,4	i2,4	i3,3
i3,4	i4,4	i1,3
i4,4,	i1,2	i4,3
i2,2	i3,2	i2,3

Top-k Buffer
i1,1.33
i2,1.33

Threshold is : 1.86

Relevance Only (RO) algorithm

IL_{u1}	IL_{u2}	IL_{u3}
i1,4	i2,4	i3,3
i3,4	i4,4	i1,3
i4,4,	i1,2	i4,3
i2,2	i3,2	i2,3

Top-k Buffer
i1,1.33
i2,1.33
i3,1.33

Threshold is : 1.73

Relevance Only (RO) algorithm

IL_{u1}	IL_{u2}	IL_{u3}
i1,4	i2,4	i3,3
i3,4	i4,4	i1,3
i4,4,	i1,2	i4,3
i2,2	i3,2	i2,3

Top-k Buffer
i1,1.33
i2,1.33
i3,1.33

Threshold is : 1.73

Relevance Only (RO) algorithm

IL_{u1}	IL_{u2}	IL_{u3}
i1,4	i2,4	i3,3
i3,4	i4,4	i1,3
i4,4,	i1,2	i4,3
i2,2	i3,2	i2,3

Top-k Buffer
i4,1.6
i1,1.33
i2,1.33
i3,1.33

Threshold is : 1.73

Relevance Only (RO) algorithm

IL_{u1}
i1,4
i3,4
i4,4,
i2,2

IL_{u2}
i2,4
i4,4
i1,2
i3,2

IL_{u3}
i3,3
i1,3
i4,3
i2,3

Top-k Buffer
i4,1.6
i1,1.33
i2,1.33
i3,1.33

Threshold is : 1.73

Relevance Only (RO) algorithm

IL_{u1}	IL_{u2}	IL_{u3}
i1,4	i2,4	i3,3
i3,4	i4,4	i1,3
i4,4	i1,2	i4,3
i2,2	i3,2	i2,3

Top-k Buffer
i4,1.6
i1,1.33
i2,1.33
i3,1.33

Threshold is : 1.73

Relevance Only (RO) algorithm

- **After 8 Sorted Accesses (3 on IL(u1), 3 on IL(u2) and 2 on IL(u3))**

IL_{U1}	IL_{U2}	IL_{U3}
i1,4	i2,4	i3,3
i3,4	i4,4	i1,3
i4,4	i1,2	i4,3
i2,2	i3,2	i2,3

Top-k Buffer
i4,1.6
i1,1.33
i2,1.33
i3,1.33

Threshold is : 1.6

IT STOPS!

Top-1 Item is i4

Enforcing monotonicity

- **Input: 3 relevance lists (IL_{u1} ,IL_{u2})**
- **… and one disagreement list $DL_{u1,u2}$**
- **Disagreement lists sorted in increasing disagreement value**

IL_{U1}	IL_{U2}	$DL_{U1,U2}$
i1,4	i2,4	i4,0
i3,4	i4,4	i3,2
i4,4	i1,2	i2,2
i2,2	i3,2	i1,2

Full Materialization (FM)

- Input: relevance lists (IL_{u1}, IL_{u2}, IL_{u3}) and 3 pair-wise disagreement lists ($DL_{u1,u2}$, $DL_{u1,u3}$, $DL_{u2,u3}$)
- getNext() accesses cursors in all lists
- Items encountered in disagreement lists play a role in pruning (when disagreement values drop considerably)

IL_{U1}	IL_{U2}	IL_{U3}	$DL_{U1,U2}$	$DL_{U1,U3}$	$DL_{U2,U3}$
i1,4	i2,4	i3,3	i4,0	i4,1	i4,1
i3,4	i4,4	i1,3	i3,2	i3,1	i1,1
i4,4	i1,2	i3,3	i2,2	i2,1	i3,1
i2,2	i3,2	i2,3	i1,2	i1,2	i2,1

Full Materialization (FM) algorithm

IL_{U1}	IL_{U2}	IL_{U3}	$DL_{U1,U2}$	$DL_{U1,U3}$	$DL_{U2,U3}$
i1,4	i2,4	i3,3	i4,0	i4,1	i4,1
i3,4	i4,4	i1,3	i3,2	i3,1	i1,1
i4,4	i1,2	i3,3	i2,2	i2,1	i3,1
i2,2	i3,2	i2,3	i1,2	i1,2	i2,1

i1,1.33

Top-k Buffer

Threshold is : 1.93

Full Materialization (FM) algorithm

IL_{u1}	IL_{u2}	IL_{u3}	$DL_{u1,u2}$	$DL_{u1,u3}$	$DL_{u2,u3}$
i1,4	i2,4	i3,3	i4,0	i4,1	i4,1
i3,4	i4,4	i1,3	i3,2	i3,1	i1,1
i4,4	i1,2	i3,3	i2,2	i2,1	i3,1
i2,2	i3,2	i2,3	i1,2	i1,2	i2,1

i1,1.33
i2,1.33

Top-k Buffer

Threshold is : 1.86

Full Materialization (FM) algorithm

IL_{u1}	IL_{u2}	IL_{u3}	$DL_{u1,u2}$	$DL_{u1,u3}$	$DL_{u2,u3}$
i1,4	i2,4	i3,3	i4,0	i4,1	i4,1
i3,4	i4,4	i1,3	i3,2	i3,1	i1,1
i4,4	i1,2	i3,3	i2,2	i2,1	i3,1
i2,2	i3,2	i2,3	i1,2	i1,2	i2,1

i1,1.33
i2,1.33
i3,1.33

Top-k Buffer

Threshold is : 1.73

Full Materialization (FM) algorithm

IL_{u1}	IL_{u2}	IL_{u3}	$DL_{u1,u2}$	$DL_{u1,u3}$	$DL_{u2,u3}$
i1,4	i2,4	i3,3	i4,0	i4,1	i4,1
i3,4	i4,4	i1,3	i3,2	i3,1	i1,1
i4,4	i1,2	i3,3	i2,2	i2,1	i3,1
i2,2	i3,2	i2,3	i1,2	i1,2	i2,1

i4,1.6
i2,1.33
i3,1.23
i4,1.33

Top-k Buffer

Threshold is : 1.73

Full Materialization (FM) algorithm

IL_{u1}	IL_{u2}	IL_{u3}	$DL_{u1,u2}$	$DL_{u1,u3}$	$DL_{u2,u3}$
i1,4	i2,4	i3,3	i4,0	i4,1	i4,1
i3,4	i4,4	i1,3	i3,2	i3,1	i1,1
i4,4	i1,2	i3,3	i2,2	i2,1	i3,1
i2,2	i3,2	i2,3	i1,2	i1,2	i2,1

i4,1.6
i2,1.33
i3,1.23
i4,1.33

Top-k Buffer

Threshold is : 1.66

Full Materialization (FM) algorithm

IL_{u1}	IL_{u2}	IL_{u3}	$DL_{u1,u2}$	$DL_{u1,u3}$	$DL_{u2,u3}$
i1,4	i2,4	i3,3	i4,0	i4,1	i4,1
i3,4	i4,4	i1,3	i3,2	i3,1	i1,1
i4,4	i1,2	i3,3	i2,2	i2,1	i3,1
i2,2	i3,2	i2,3	i1,2	i1,2	i2,1

After 6 Sorted Accesses(1 on each list)

i4,1.6
i2,1.33
i3,1.23
i4,1.33

Top-k Buffer

Threshold is : 1.6

Score(i4) = Threshold = 1.6

IT STOPS!

Top-1 item is i4

Full Materialization (FM)

- **Proliferation of disagreement lists**

IL_{u1}	IL_{u2}	IL_{u3}	$DL_{u1,u2}$	$DL_{u1,u3}$	$DL_{u2,u3}$
i1,4	i2,4	i3,3	i4,0	i4,1	i4,1
i3,4	i4,4	i1,3	i3,2	i3,1	i1,1
i4,4	i1,2	i3,3	i2,2	i2,1	i3,1
i2,2	i3,2	i2,3	i1,2	i1,2	i2,1

FM space overhead

Conservative example:

- *70K users, 10K items*
- *14 trillion entries in pair-wise disagreement lists*
- ***2 Terabyte of storage!***

Don’t try this at home either!

Outline

- ✓ Intro
- ✓ Problem definition
- ✓ Top-k applicability
- **Performance optimizations**
  - Behavior factoring
  - Partial materialization
  - Threshold sharpening
- **Experiments**

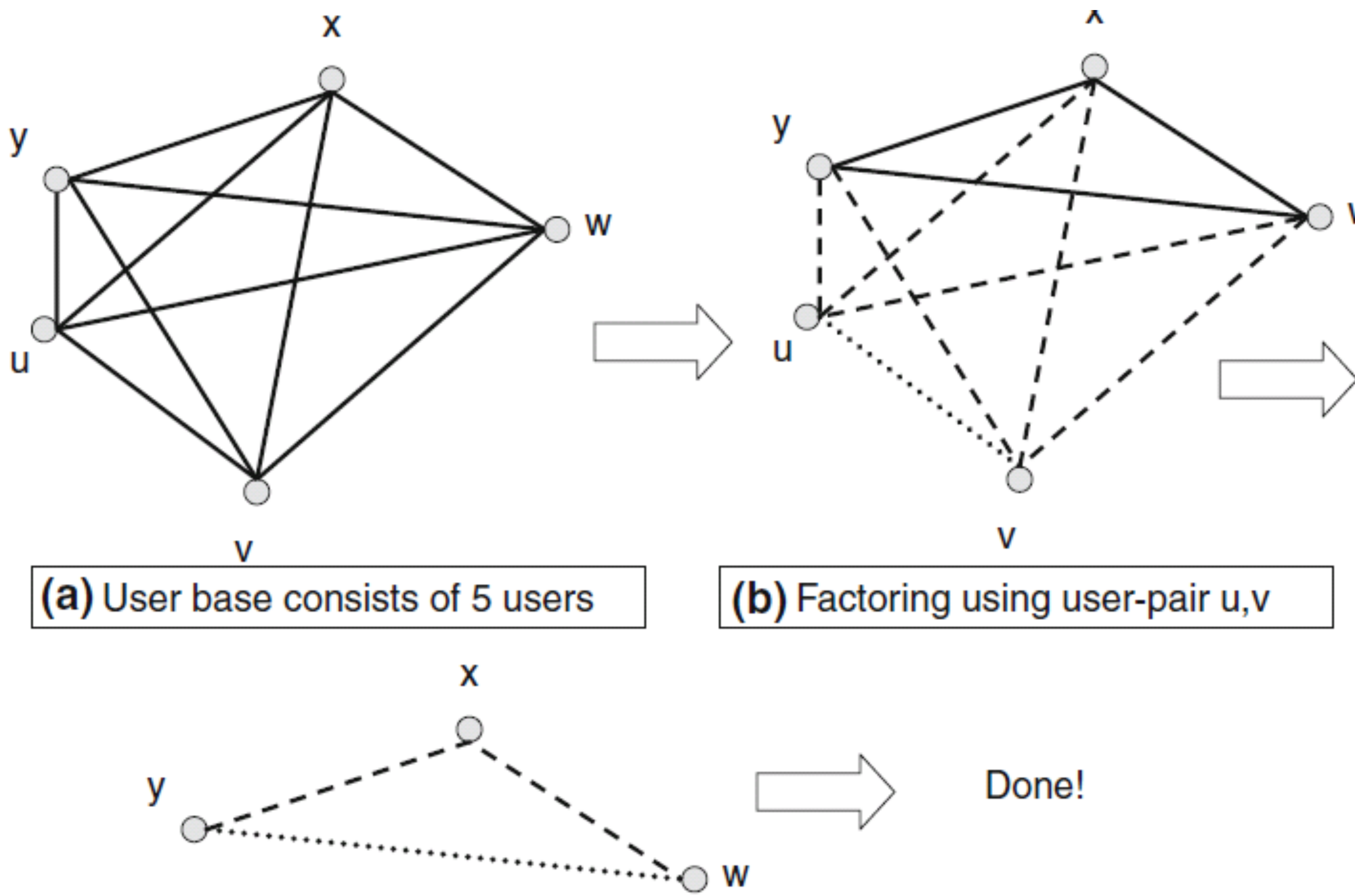
Optimizations

- **Behavior Factoring**
  - store shared disagreement only once
  - does not always reach space budget
- **Partial Materialization**
  - given a space budget, which m out of n(n-1)/2 disagreement lists, to materialize?
- **Threshold Sharpening**
  - exploit the dependencies between relevance and disagreement lists and sharpen thresholds in FM, RO and PM algorithms?

Behavior Factoring

- **Intuition: If two users *u* and *v* agree on a set of items *S*, their lists *DL(u,w)* and *DL(v,w)* with any other user *w* share the same values for S.**
- **Store DL(S,w) once**
- **Overall space reduce by size of S**
- **Redefine getNext() to work on both disagreement lists and factored out lists**
- **Virtually, no impact on performance**
- **Does not always guarantee fitting into a space budget**

Factoring steps



Why Partial Materialization ?

- A set of 10,000 users has 49995000 disagreement lists
- Only 10% of the disagreement lists can be materialized, given a space budget
- Problem : Which 4999500 lists should we choose so that those gives “maximum benefit” during query processing?
- Intuition : Materialize only those lists that significantly improves efficiency.
- Recommendation Algorithm needs to be adapted to it (referred to as PM in the paper)

Partial Materialization (PM)

- **Problem: which lists should we choose so that those give "maximum benefit" during query processing?**

- **Intuition:**
  - overall performance is a balance between the total number of distinct items which need to be processed and the number of SAs
  - If none of top items in DL2 is in final output, every SA on DL2 is overhead → best not to materialize DL2

Partial materialization without factoring

- **Determine the subset of pairs *M* ⊆ *S* s.t. |*M*| = *m/r* and *tM* = *G* ⊆ *U* *p(G) tM(G)* is minimized.**

- **Solution**
  - Group query *G* will two users,
  - *p(G)* is reliably known for all pairs of users *G*.
  - Avoid examining all user pairs for any user pair *(u, v)*,
    p({u, v}) = |{*Gi* |*u, v* ∈ *Gi* }|

Partial materialization after factoring

- To identify the subset of the factored as well as common component of the original disagreement list for each pair is materialized.
- Disagreement lists have already been factored.

- Determine the subset of pairs *M* ⊆ *S* s.t. the space required by all factored and common lists corresponding to all pairs in *M* is at most *m*, and *tM* = *G* ⊆ *U p(G) tM(G)* is minimized.

$$Space(S') = \sum_{P_i \in S'} |\mathcal{DL}_{S(P_i)}| + \sum_{\mathcal{DL}_C \in C(S')} |\mathcal{DL}_C|$$

PM algorithm

Adaptation of the ½-approx algorithm for 0/1 Knapsack Problem

Sort the table on decreasing difference (#SAs) and consider first m rows

User Pair	#SAs without disagreement list	#SAs with disagreement lists	Difference in #SAs
$\{U_1,U_2\}$	200	100	100
$\{U_3,U_4\}$	290	195	95
$\{U_{10},U_9\}$	170	100	70
$\{U_6,U_7\}$	230	190	40
$\{U_2,U_3\}$	175	145	30
$\{U_5,U_6\}$	200	179	21
$\{U_7,U_8\}$	120	100	20
--	--	--	--
--	--	--	--

<= *m*

Threshold sharpening

IL_{u1}
i1,0.5
i3,0.5

IL_{u2}
i2,0.5
i3, 0.4

$DL_{u1,u2}$
i3,0.2
I1,0.3

Threshold = 1.3

Maximize

$$(i_{U1} + i_{U2})/2 + (1 - |i_{U1} - i_{U2}|)$$

s.t.

$$0 <= i_{U1} <= 0.5$$

$$0 <= i_{U1} <= 0.5$$

$$0.2 <= | i_{U1} - i_{U2} | <= 1$$

New Threshold = 1.2

Outline

- ✓ Intro
- ✓ Problem definition
- ✓ Top-k applicability
- ✓ Performance optimizations
- Experiments

Experiments

- **Dataset**
  - MovieLens data set
  - 71,567 users, 10,681 movies, 10,000,054 ratings

- **Performance Experiments**
  - ➢ Dynamic Computation with Predicted Rating List Only (RO),
  - ➢ Full Materialization (FM)
  - ➢ Partial Materialization

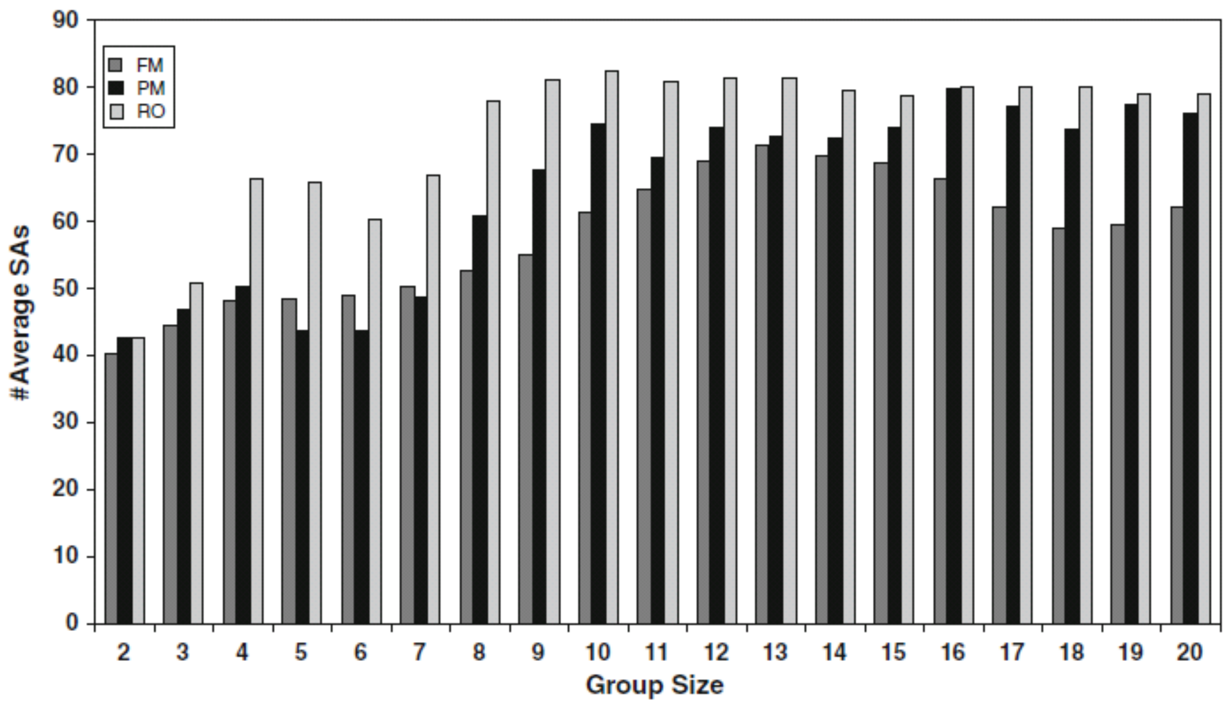
  - Performance (#SAs) comparison of FM, RO and PM varying group size, similarity and k.
  - Effectiveness of behavior factoring, partial materialization and threshold sharpening

Group recommendation algorithms

- **The Full Materialization (FM) Algorithm**
  - *IL* of each user in the input group *G* and disagreement lists *DL* for every pair of users in *G.*
- **The Ratings Only (RO) Algorithm**
  - Only when the predicted rating lists are present and none of the *DLs* are available.
  - Consume less space.
- **The Partial Materialization (PM) Algorithm**
  - Some disagreement lists are materialized,

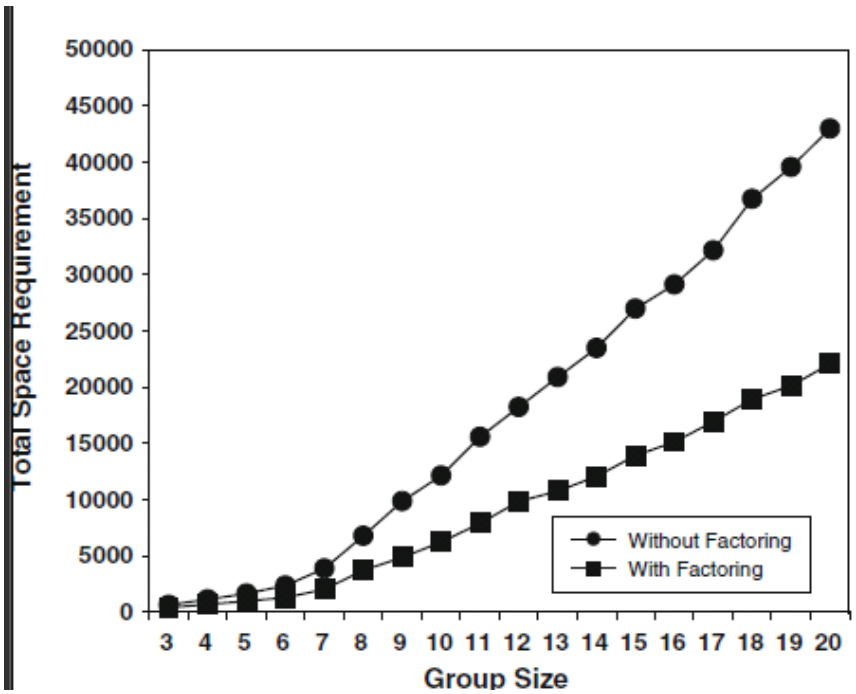
Space reduction techniques and their impact on query processing

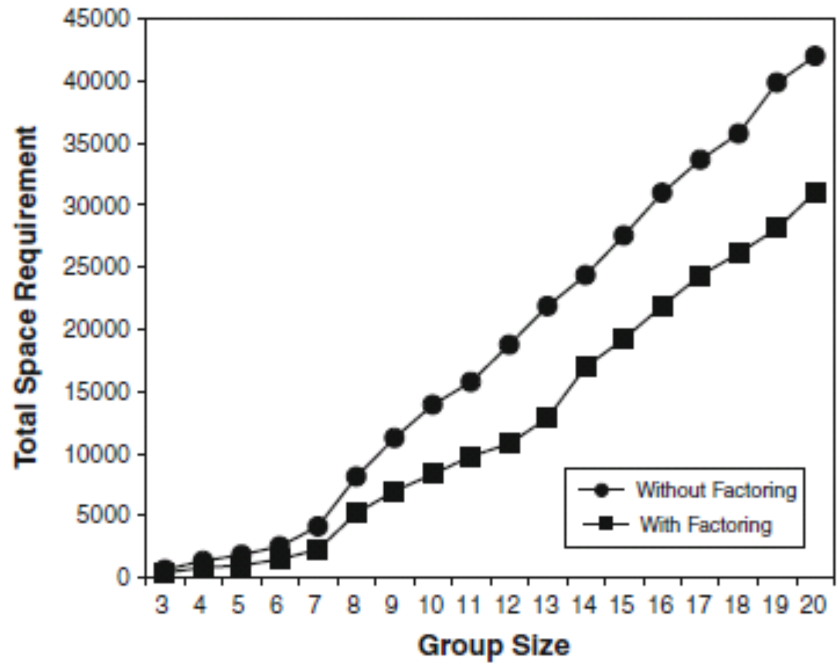
- FM gets better as group size is increased
- RO performs the worst among all three in all cases
- PM is the best solution



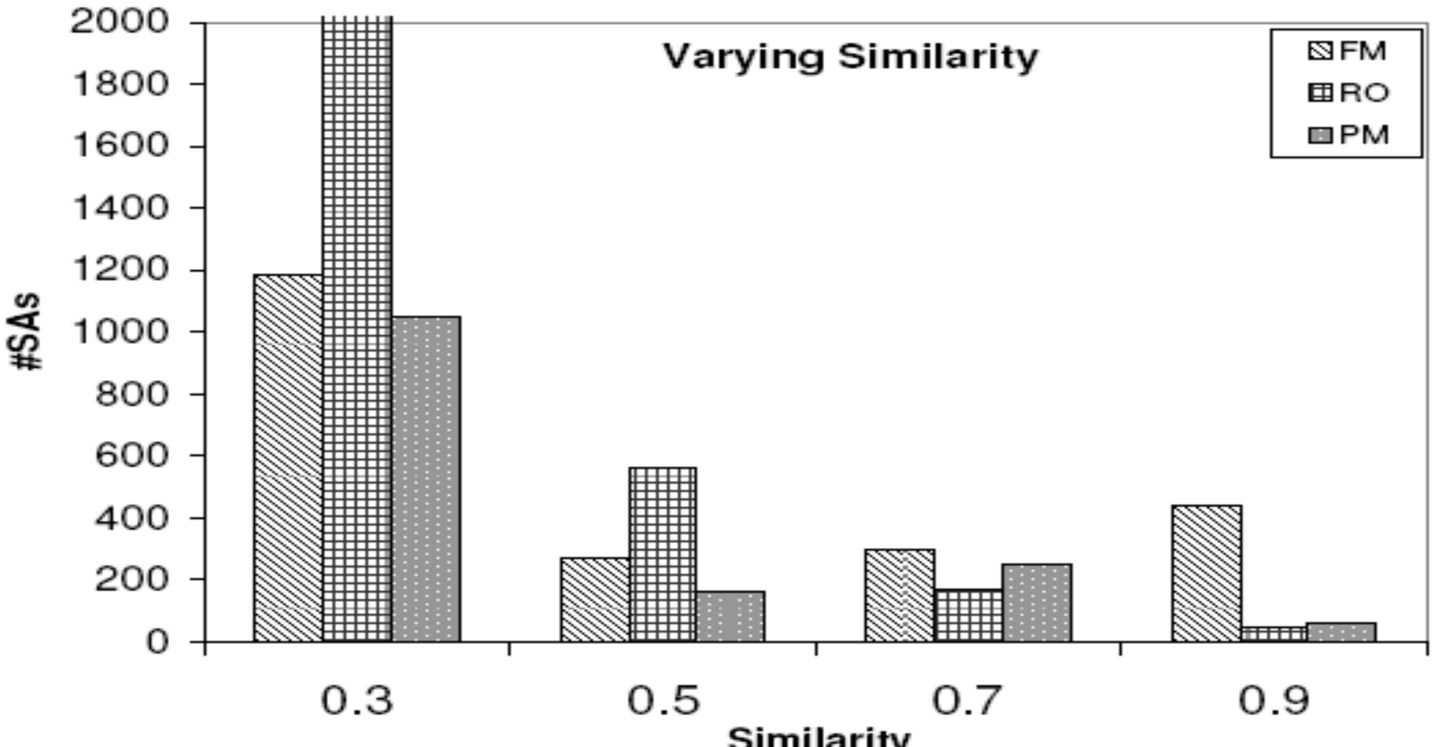
Space reduction techniques and their impact on query processing

Factoring algorithm is effective and performs well





Performance results

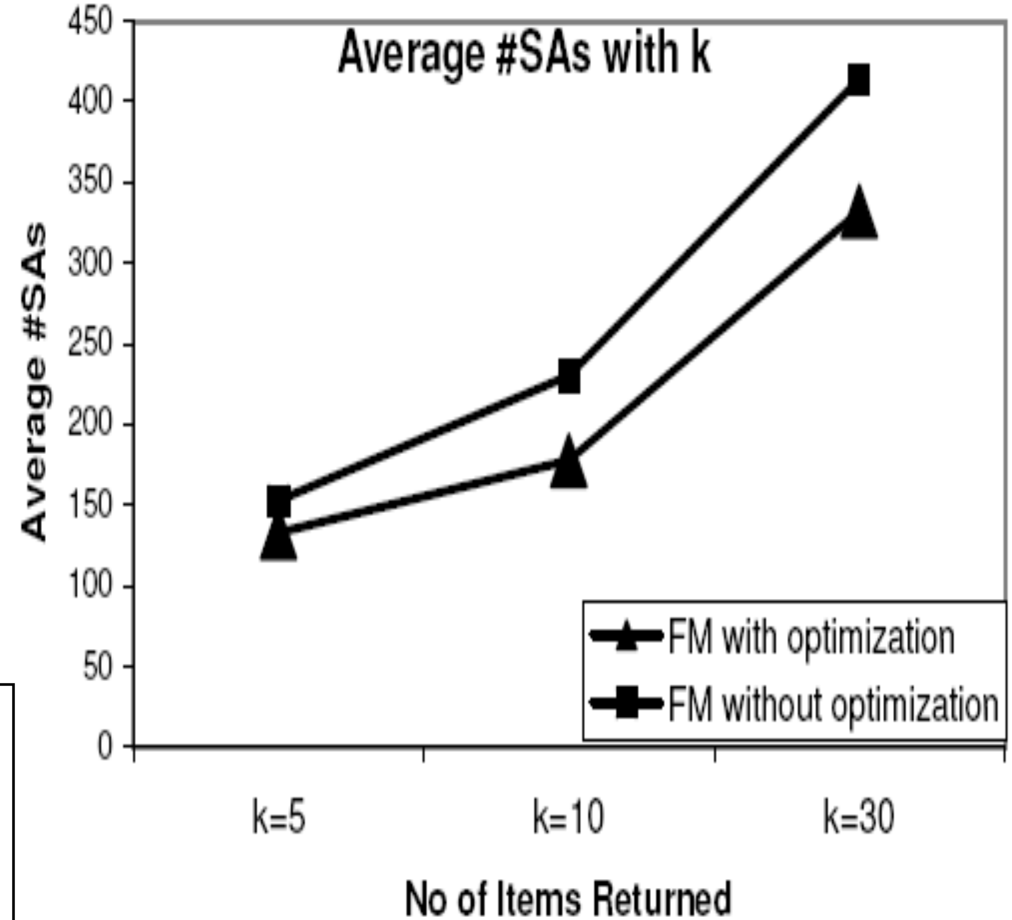


•***Less sorted accesses (SAs)*** *are required for more* ***similar user groups***

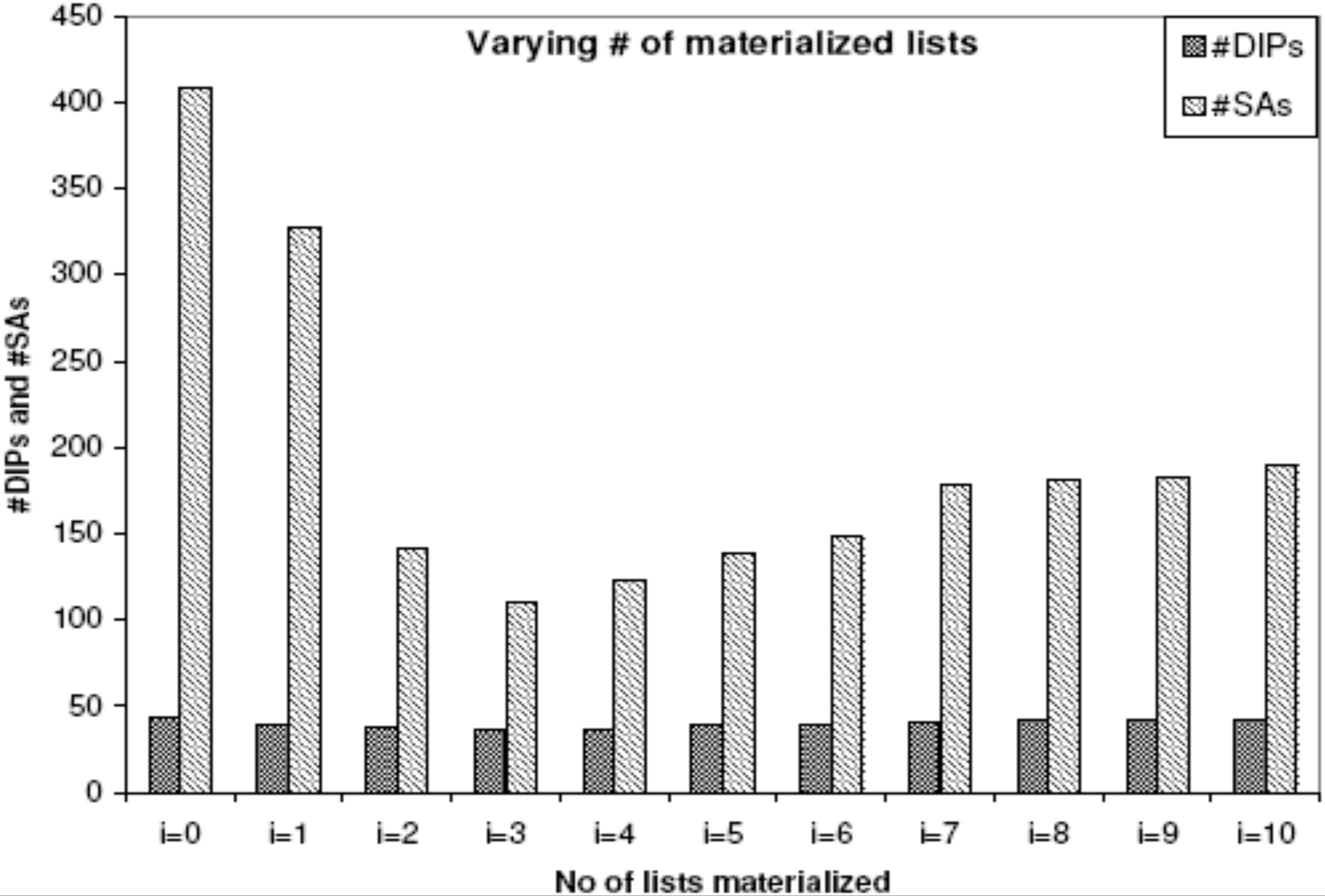
•***Disagreement lists*** *are important for* ***Dissimilar user groups***

•***FM*** *is the best performer for very dissimilar user groups,* ***RO*** *is the best algorithm for very similar user groups.*

Performance results



Optimization during threshold calculation always achieves better performance (less #SAs) than without optimization case.



Sometimes only few disagreement lists attain the best performance. Therefore ***Partial Materialization*** *is important*

Summary and outlook

- **Recommendations to *ad-hoc groups* will become more important**
  - think Google+
- **Efficient group recommendation**
  - maintaining disagreement lists enables efficient top-k processing
  - threshold sharpening optimizes response time
  - behavior factoring and partial materialization reduce index size
  - full materialization does not always perform better than partial materialization ➔ potential for new optimization problem
- **Next lecture**
  - How do we measure *answer quality* and *user satisfaction*?

References and further reading

1. *Group Recommendation: Semantics and Efficiency.*
   Sihem Amer-Yahia, Senjuti Roy, Ashish Chawla, Gautam Das, Cong Yu. VLDB 2009.
2. *Space Efficiency in Group Recommendation.*
   Senjuti Roy, Sihem Amer-Yahia, Ashish Chawla, Gautam Das, Cong Yu. VLDB J. 2010.
3. *Group recommendation system for Facebook*.
   Enkh-Amgalan Baatarjav, Santi Phithakkitnukoon, Ram Dantu. OTM Workshops, 2008.
4. A *group recommendation system with consideration of interactions among group members.* Yen-Liang Chen and Li-Chen Cheng and Ching-Nan Chuang**.** ESWA 2008.
5. *Case-based group recommendation: Compromising for success.*
   Kevin McCarthy, Lorraine McGinty, and Barry Smyth. ICCBR, 2007.
6. *PolyLens: A recommender system for groups of users.*
   Mark O'Connor, Dan Cosley, Joseph A. Konstan, John Riedl. ECSCW, 2001.
7. *Restaurant recommendation for group of people in mobile environments using probabilistic multi-criteria decision making*.
   Moon-Hee Park and Han-Saem Park and Sung-Bae Cho. APCHI 2008.

Questions?