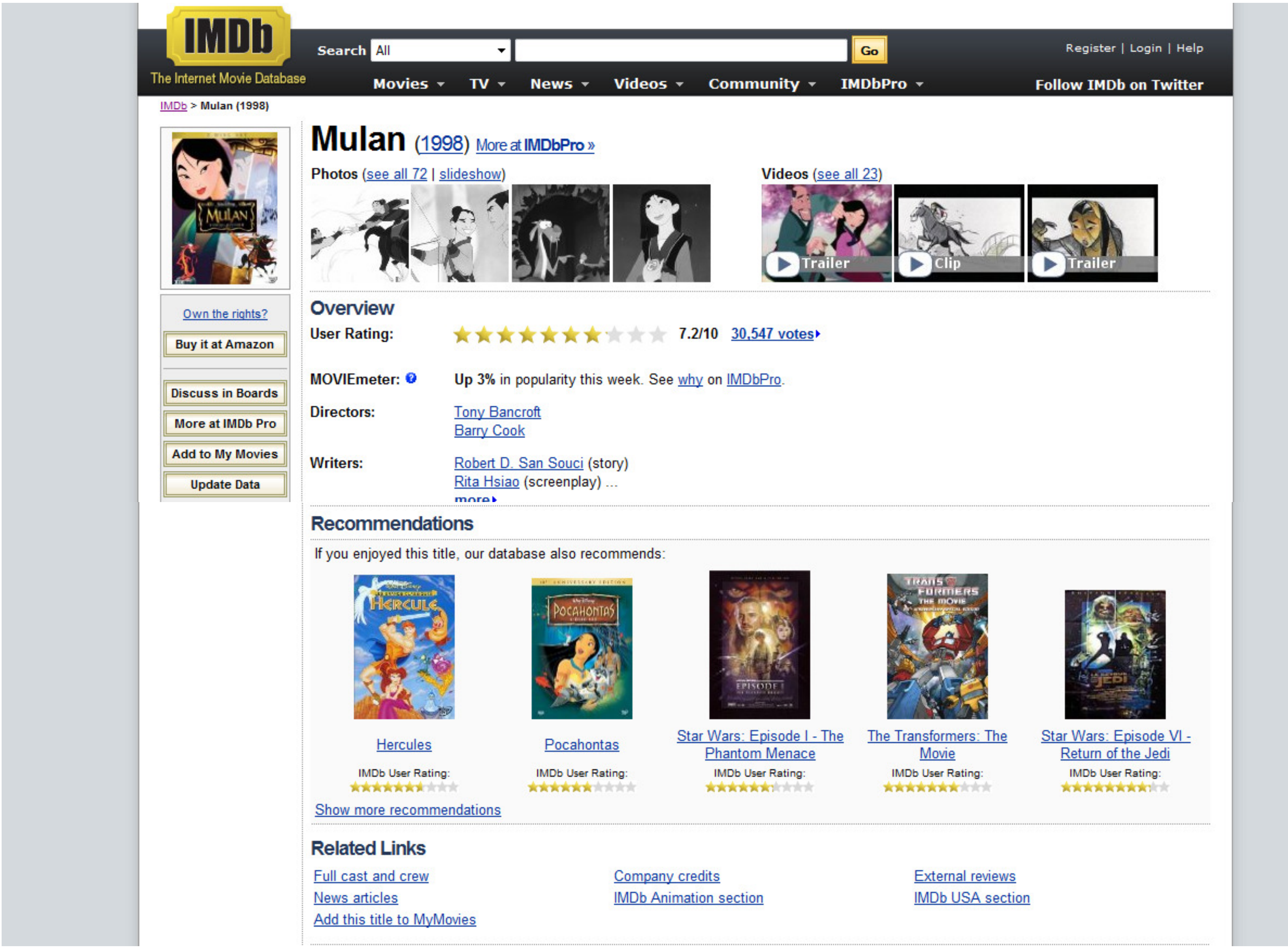
Group Recommendation:

Semantics and Efficiency

A well-known Example (Amazon)

Another well-known Example (IMDb)



Recommendations: Application Field

- Sales

  (recommend items that the user might be interested to buy)

- Entertainment

  (recommend a good movie or a good restaurant)

Motivation for group recommendations

- In most cases: recommendations for a single user

- But: for entertainment purposes the users often form groups (movies, restaurants)

- How can we create good recommendations for a group of users?

Outline

- Recommendation for individuals
- Group recommendation
- Experiments
- Conclusion

Outline

- Recommendation for individuals
- Group recommendation
- Experiments
- Conclusion

Individual Recommendation: Item based

- In order to estimate the user’s opinion about an new item i, use items i’ similar to the user’s previously highly rated items

$$relevance(u,i) = \sum_{i' \in I} ItemSim(i,i') \times rating(u,i')$$

- I: set of all items
- u: current user
- ItemSim(i,i’): How similar i,i’ are (eg.cosine similarity)

Individual Recommendation: User based (Collaborative filtering)

- In order to estimate the user's u opinion about an new item i, use the opinion of people u', who share the user's u interests, for this item

$$relevance(u,i) = \sum_{u' \in U} UserSim(u,u') \times rating(u',i)$$

- U: set of users
- UserSim(u,u'): How similar u, u' are

Outline

- Recommendation for individuals
- Group recommendation
- Experiments
- Conclusion

Group Recommendation

- Basic approaches
- The consensus function
- The TA Algorithm
- The monotonicity issue
- Full and Partial Materialization Algorithms
- Sharpening the Thresholds

Basic Approaches

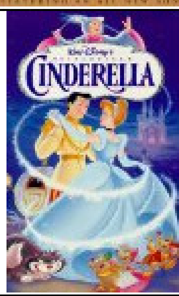
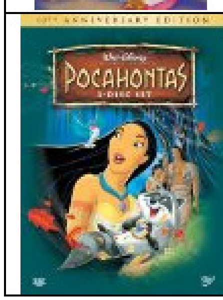
- Preference Aggregation

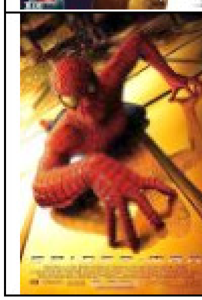
  group members’ prior ratings are aggregated into a single virtual user profile. Recommendations are made for the virtual user

- Score Aggregation

  each member’s individual recommendations are computed and merged into a single list for the group, where the score of each item is aggregated from individual recommendations

Preference Aggregation (Example)

Known Movie	Mary	Chris	Helen	Virtual User
	0.7	0.2	0.5	0.46
	0.6	0.5	0.6	0.56
	0.3	0.7	0.9	0.63

Unseen Movies	Virtual User
	0.7
	0.9
	0.3

Score Aggregation Functions

- Average

  maximize the average of group members' scores for an item.

  But: ignores "veto" votes

- Least Misery

  maximize the lowest score among all group members.

  But: misses the item that is liked by all members except one

Score Aggegation (Example)

	Mary	Chris	Helen	Avg	Min
	1.0	1.0	0.0	**0.66**	0.0
	0.4	0.8	0.4	0.53	**0.4**
	0.4	0.2	0.6	0.4	0.2

Consensus Function (1)

Better:

- For each item calculate two components:
  - Relevance $rel(G, i)$
  - Disagreement $dis(G,i)$

Consensus Function (2)

- Group Relevance:
  - Average: $$rel(G,i) = \frac{1}{|G|}\sum_{u \in G} relevance(u,i)$$
  - Least Misery: $$rel(G,i) = Min(relevance(u,i))$$
- Group Disagreement:
  - Average Pair-wise Disagreement:

$$dis(G,i) = \frac{2}{|G|(|G|-1)}\sum_{(u,v) \in G}(|relevance(u,i) - relevance(v,i)|)$$

  - Disagreement Variance:

$$dis(G,i) = \frac{1}{|G|}\sum_{u \in G}(relevance(u,i) - mean)^2$$

Consensus Function (3)

- So the Consensus Function will be:

$$F(G,i) = w_1 \times rel(G,i) + w_2 \times (1 - dis(G,i))$$
$$where \;\; w_1 + w_2 = 1.0$$

- But: …

Consensus Function (example)

	Mary	Chris	Helen	Avg	Δ(M,C)	Δ(C,H)	Δ(H,M)	ΣΔ	CF(0.5)
	1.0	1.0	0.0	0.66	0.0	1.0	1.0	2	0.5
	0.4	0.8	0.4	0.53	0.4	0.4	0.0	0.8	**0.625**
	0.4	0.2	0.6	0.4	0.2	0.4	0.2	0.8	0.56

Q: How to calculate the CF?
A: TA algorithm

- Input:

  for each user a list of items, sorted in descending score order (relevance order), together with the score for each item

- Output:

  the top-k items according to some aggregating function

TA algorithm

while there are items in lists

Scan lists;

Consider item I at position pos_i list L_i;

$high_i = s(u_i, I)$;

if $I \notin top-k$ then

  look up $s_v(I)$ in all lists L_v with $v \neq i$;

  $score(I) = aggr\{s_v(I), v = 1...m\}$;

if $score(I) > \min-k$ then

  add I to $\text{top}-\text{k}$ and remove $\text{min}-\text{score}\ I'$;

$threshold = aggr\{high_v, v = 1...m\}$;

if $threshold \leq \min-k$ then exit;

TA in action (k=2, aggr=Sum)- in general

- Depth=1

U1	U2	U3	Threshold
I78: 0.9	I64: 0.9	I10: 0.7	2.5
I23: 0.8	I23: 0.6	I78: 0.5	
I10: 0.8	I10: 0.6	I64: 0.3	
I1: 0.7	I12: 0.2	I99: 0.2	
I88: 0.2	I78: 0.1	I34: 0.1	

Rank	ItemID	Score
1	I78	0.9

TA in action (k=2, aggr=Sum)- in general

- Depth=1

U1	U2	U3	Threshold
I78: 0.9	I64: 0.9	I10: 0.7	2.5
I23: 0.8	I23: 0.6	I78: 0.5	
I10: 0.8	I10: 0.6	I64: 0.3	
I1: 0.7	I12: 0.2	I99: 0.2	
I88: 0.2	I78: 0.1	I34: 0.1	

Rank	ItemID	Score
1	I78	1.5

TA in action (k=2, aggr=Sum)- in general

- Depth=1

U1	U2	U3	Threshold
I78: 0.9	I64: 0.9	I10: 0.7	2.5
I23: 0.8	I23: 0.6	I78: 0.5	
I10: 0.8	I10: 0.6	I64: 0.3	
I1: 0.7	I12: 0.2	I99: 0.2	
I88: 0.2	I78: 0.1	I34: 0.1	

Rank	ItemID	Score
1	I78	1.5
2	I64	1.2

TA in action (k=2, aggr=Sum)- in general

- Depth=1

U1	U2	U3	Threshold
I78: 0.9	I64: 0.9	I10: 0.7	2.5
I23: 0.8	I23: 0.6	I78: 0.5	
I10: 0.8	I10: 0.6	I64: 0.3	
I1: 0.7	I12: 0.2	I99: 0.2	
I88: 0.2	I78: 0.1	I34: 0.1	

Rank	ItemID	Score
1	I10	2.1
2	I78	1.5

TA in action (k=2, aggr=Sum)- in general

- Depth=2

U1	U2	U3	Threshold
I78: 0.9	I64: 0.9	I10: 0.7	2.5
I23: 0.8	I23: 0.6	I78: 0.5	1.9
I10: 0.8	I10: 0.6	I64: 0.3	
I1: 0.7	I12: 0.2	I99: 0.2	
I88: 0.2	I78: 0.1	I34: 0.1	

Rank	ItemID	Score
1	I10	2.1
2	I78	1.5

TA in action (k=2, aggr=Sum)- in general

- Depth=3

U1	U2	U3	Threshold
I78: 0.9	I64: 0.9	I10: 0.7	2.5
I23: 0.8	I23: 0.6	I78: 0.5	1.9
I10: 0.8	I10: 0.6	I64: 0.3	1.7
I1: 0.7	I12: 0.2	I99: 0.2	
I88: 0.2	I78: 0.1	I34: 0.1	

Rank	ItemID	Score
1	I10	2.1
2	I78	1.5

TA in action (k=2, aggr=Sum)- in general

- Depth=4 (Score(I78)<Threshold: STOP)

U1	U2	U3	Threshold
I78: 0.9	I64: 0.9	I10: 0.7	2.5
I23: 0.8	I23: 0.6	I78: 0.5	1.9
I10: 0.8	I10: 0.6	I64: 0.3	1.7
I1: 0.7	I12: 0.2	I99: 0.2	1.1
I88: 0.2	I78: 0.1	I34: 0.1	

Rank	ItemID	Score
1	I10	2.1
2	I78	1.5

The monotonicity issue (1)

- The correct early stopping of TA Algorithm is possible only when the aggregating function is monotone
- The consensus function is comprised by two components: group relevance and group disagreement. Both of them should be monotone.
- Relevance (average, min) is monotone
- Disagreement: ?

The monotonicity issue (2)

- Group Disagreement is not monotonic w.r.t. relevance lists

Rel(U1)	Rel(U2)
I1: 0.4	I1: 0.3
I2: 0.3	I2: 0.3

Rel(U1)	Rel(U2)
I2: 0.3	I1: 0.4
I1: 0.4	I2: 0.4

$$dis(G,i) = \frac{2}{|G|(|G|-1)} \sum_{(u,v)\in G} (|relevance(u,i) - relevance(v,i)|)$$

The monotonicity issue (2)

- Group Disagreement is not monotonic w.r.t. relevance lists

Rel(U1)	Rel(U2)
I1: 0.4	I1: 0.3
I2: 0.3	I2: 0.3

Rel(U1)	Rel(U2)
I2: 0.3	I1: 0.4
I1: 0.4	I2: 0.4

- Solution: maintain also pairwise disagreement lists

$$dis(G,i) = \frac{2}{|G|(|G|-1)} \sum_{(u,v)\in G} (|relevance(u,i) - relevance(v,i)|)$$

TA for Group Recommendation

- Relevance list for each user, with items sorted in descending order
- Disagreement lists for each pair of users, with items sorted in ascending order
- Aggregation Function = Consensus Function

Rel(u1,i)	Rel(u2,i)	Dis(u1,u2,i)
I1: 0.5	I2: 0.5	I2: 0.0
I2: 0.5	I4: 0.4	I4: 0.1
I3: 0.4	I1: 0.3	I5: 0.1
I4: 0.3	I6: 0.3	I6: 0.1
I5: 0.3	I7: 0.3	I1: 0.2
I6: 0.2	I3: 0.2	I3: 0.2
I7: 0.1	I5: 0.2	I7: 0.2

Group Recommendation Algorithm with Fully Materialized Disagreement Lists

```
Require: Group G, consensus function F;
Retrieve relevance lists IL_u for each user u in group G;
Retrieve disagreement lists DL_(u,v) for each user pair (u, v) in group G;
Cursor cur = getNext() moves across each relevance and disagreement lists;
while (cur <> NULL) do
Get entry e = (i, r) at cur;
if not(inHeap(topKHeap, e)) then
if (ComputeMaxScore(e.i, e.r, F) ≥ topKHeap.kthscore) then
ComputeExactScore; Probe ILs to compute exact scoreof e using F;
topKHeap.addToHeap(e.i, score);
else break;
end if
end if
cur = getNext();
end while
return topKList(topKHeap);
```

FM Algorithm: ConputeExactScore

- ComputeExactScore: random access (RA) on all other relevance lists to compute the score of an item i, using the input consensus function F.

$$F(G,i) = w_1 \times \frac{1}{|G|}\sum_{u \in G} r_u + w_2 \times (1 - \frac{2}{|G|(|G|-1)}\sum_{(u,v)\in G} \Delta_{u,v})$$

- DLs are not necessary to compute the final result (disagreement can be computed from relevances). They are only used to compute the threshold (using ComputeMaxScore) and hopefully, enable early termination.

FM Algorithm: ComputeMaxScore

- ComputeMaxScore produces a new threshold value at each round, in order to provide an upper bound for the score of any item that has not yet been seen by the algorithm:

if r_u is the last relevance value, read on list IL_u for all $u \in G$, and $\Delta_{u,v}$ the last pairwise disagreement value, read on disagreement list $DL_{u,v}$ for all $u, v \in G$:

$$F(G,i) \leq w_1 \times \frac{1}{|G|} \sum_{u \in G} r_u + w_2 \times \left(1 - \frac{2}{|G|(|G|-1)} \sum_{u,v \in G} \Delta_{u,v}\right)$$

RO Algorithm (Relevance lists Only)

- None of DLs available
- Consume less space
- No impact on ComputeExactScore
- But ComputeMaxScore: less tight threshold

$$F(G,i) \leq w_1 \times \frac{1}{|G|} \sum_{u \in G} r_u + w_2$$

PM Algorithm (Partial Materialization)

- Only some of the DLs are materialized
- Less space consumption than FM
- Let M be the set of all pairs of users for which disagreement lists have been materialized. Threshold:

$$F(G,i) \le w_1 \times \frac{1}{|G|}\sum_{u \in G} r_u + w_2 \times (1 - \frac{2}{|G|(|G|-1)} \sum_{(u,v)\in M} \Delta_{u,v})$$

- But…

More DLs $\Rightarrow$ Faster Algorithm?

- If none of the top items in a DL are in the end in the top-k, each SA on this DL is pure overhead

- A DL is providing the chance to tighten the threshold only if there is some skew in its items. DLs for users, that have very similar or dissimilar relevances for all items, are not useful.

- Which DLs should be materialized as a preprocessing step?

DL Materialization

- Assume only groups of pairs of users
- Assume that we know the distribution p(G), i.e., the probability that a given user group G will be queried next (query log? New users?) and prune the most unlikely groups
- Run top-k algorithm both for FM and RO and calculate the times $t_{DL_{(u,v)}}(\{u,v\})$ and $t_{\emptyset}(\{u,v\})$
- Eliminate groups for which $t_{DL_{(u,v)}}(\{u,v\}) \geq t_{\emptyset}(\{u,v\})$
- Select the set M of user pairs, so that it will maximize:

$$\sum_{(u,v)\in M} p(\{u,v\}) \cdot (t_{\emptyset}(\{u,v\}) - t_{DL_{(u,v)}}(\{u,v\}))$$

- Once the DLs are materialized, they can be used at query processing time for bigger groups

DLs: A Chance for Sharpening the Thresholds

Rel(u)	Rel(v)	Dis(u,v)
…	…	…
I1: 0.5	I2: 0.5	I3: 0.2
i_u	i_v	$\lvert i_u - i_v \rvert$

- Thresholds from each list:

$$0 \le i_u \le 0.5$$
$$0 \le i_v \le 0.5$$
$$0.2 \le \lvert i_u - i_v \rvert \le 1$$

- ComputeMaxScore: $F(G,i) \le (0.5+0.5)/2+(1-0.2/1)=1.3$

- A tighter threshold: $F(G,i) \le (0.5+0.3)/2+(1-0.2/1)=1.2$

- But…

Outline

- Recommendation for individuals
- Group recommendation
- Experiments
- Conclusion

Experiments for quality evaluation

- Data Set: MovieLens 10M ratings data set (71,567 users, 10,681 movies, 10,000,054 ratings, scale rating: 0-5)
- Relevance (individual recommendation): Collaborative filtering
- Similarity measure:

$$sim(u,u') = \frac{\left|\{i,\ i \in I_u \wedge i \in I_{u'} \wedge \left|rating(u,i) - rating(u',i)\right| \leq 2\}\right|}{\left|\{i,\ i \in I_u \vee i \in I_{u'}\}\right|}$$

Recommendation mechanisms examined

- Average Relevance Only (AR)
- Least-Misery Relevance Only (MO)
- Consensus with Pairwise Disagreement (RP)
  - RP20 where disagreement weight=0.2
  - RP80 where disagreement weight=0.8
- Consensus with Disagreement Variance (RV)
  - RV20 where disagreement weight=0.2
  - RV80 where disagreement weight=0.8

User Collection Phase - Preferences Collection

- Recruit users from Amazon Mechanical Turk
- Create a *Popular Set* of movies (top-40 in terms of popularity) and a *Diversity Set* (top-20 in terms of user rating variance, but also within the top-200 in terms of popularity)
- Create 2 HITs, each with 40 movies and 50 users :
  - Similar HIT (40 movies from Popular Set)
  - Dissimilar HIT (20 movies from Popular and 20 from Diversity)

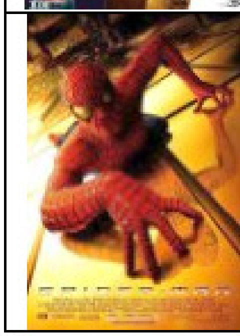
User Collection Phase – Group Formation

- Vary *Group size*: Small (3 persons) and large (8 persons) groups
- Vary *Group Cohesiveness:*
  - Similar: Users from Similar HIT and maximum summation of pairwise similarities
  - Dissimilar: Users from Dissimilar HIT and minimum summation of pairwise similarities
  - Random

Group Judgement Phase

- For each group generate group recommendations
- For each member of group generate individual recommendations with relevance values for each movie
- Create Group HIT: given, for each group member, the relevance value of each movie in the Group Recommendations Set, decide if the movie is a good recommendation

Group judgement phase (example)

Unseen movies	Estim.Rating for User 1	Estim.Rating for User 2	Estim.Rating for User 3	Will you recommend it?
	0.8	0.7	0.9	1
	0.7	0.9	0.9	1
	0.8	0.3	0.9	0

The Quality Measure

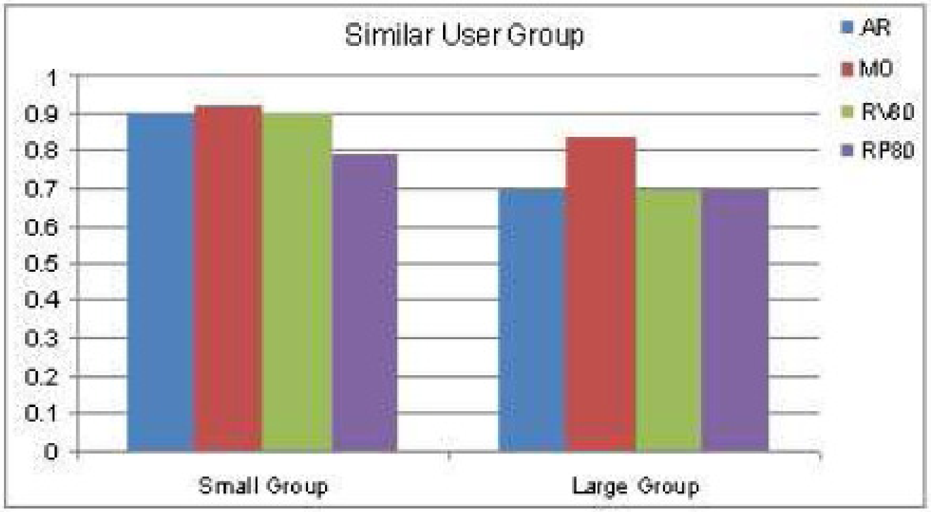
- To evaluate the recommendation strategies, we use the Discounted Cumulative Gain (DCG)
- For a 10-movie recommendation list:

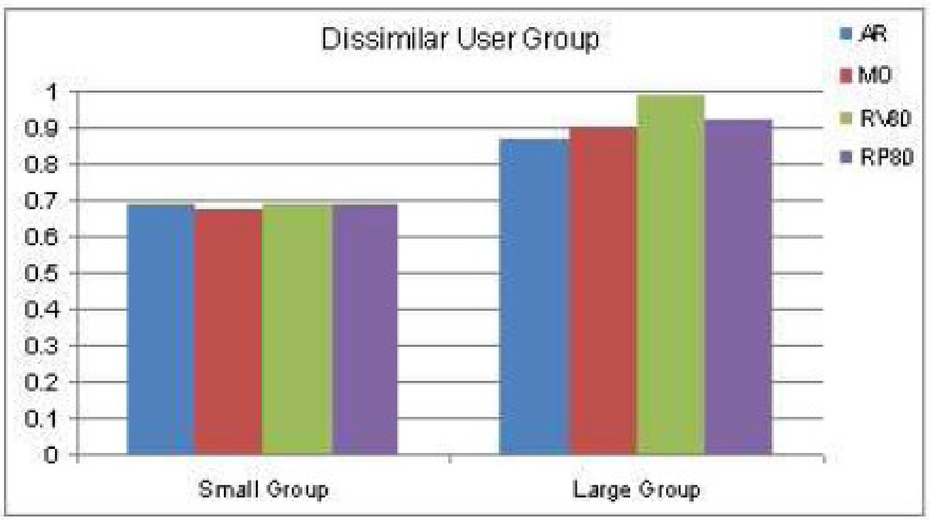
$$DCG_{10} = rel_1 + \sum_{i=2}^{10} \frac{rel_i}{\log_2(i)}$$

- If item at position I is a good recommendation, then $rel_i = 1$ else $rel_i = 0$

- Normalize: $$NDCG_{10} = \frac{DCG_{10}}{DCG_{10}(perfect\ result)}$$

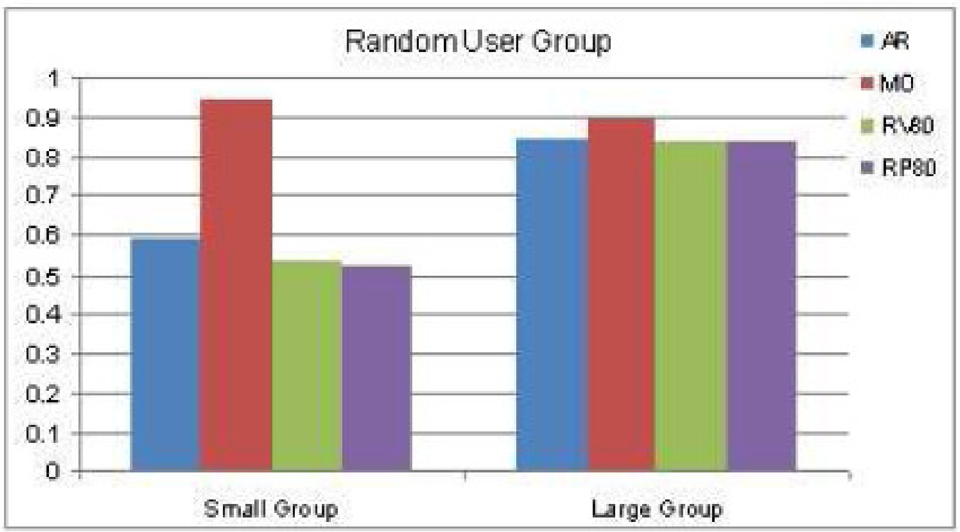
Quality Results (1)





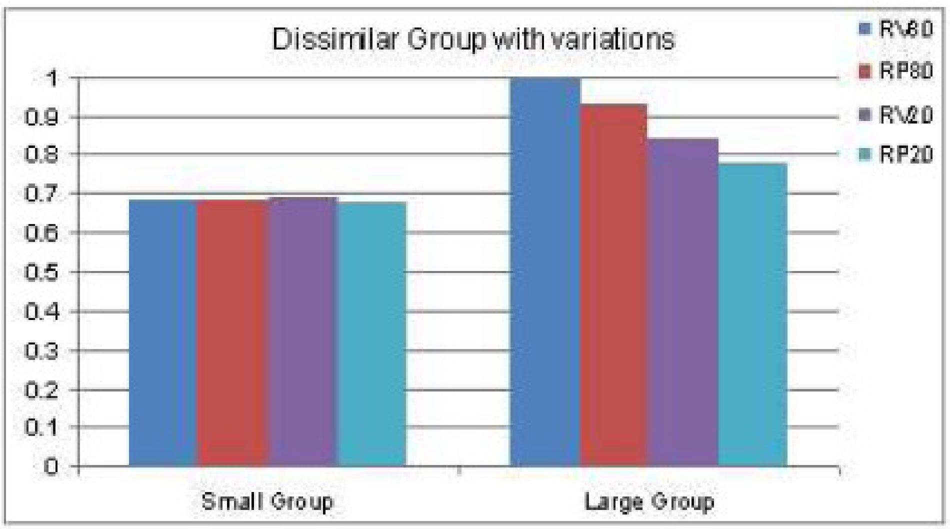
Similarities for User Study Groups		
	Small	Large
Similar	0.89	0.90
Dissimilar	0.29	0.27
Random	0.69	0.73

Quality Results (2)



Similarities for User Study Groups		
	Small	Large
Similar	0.89	0.90
Dissimilar	0.29	0.27
Random	0.69	0.73

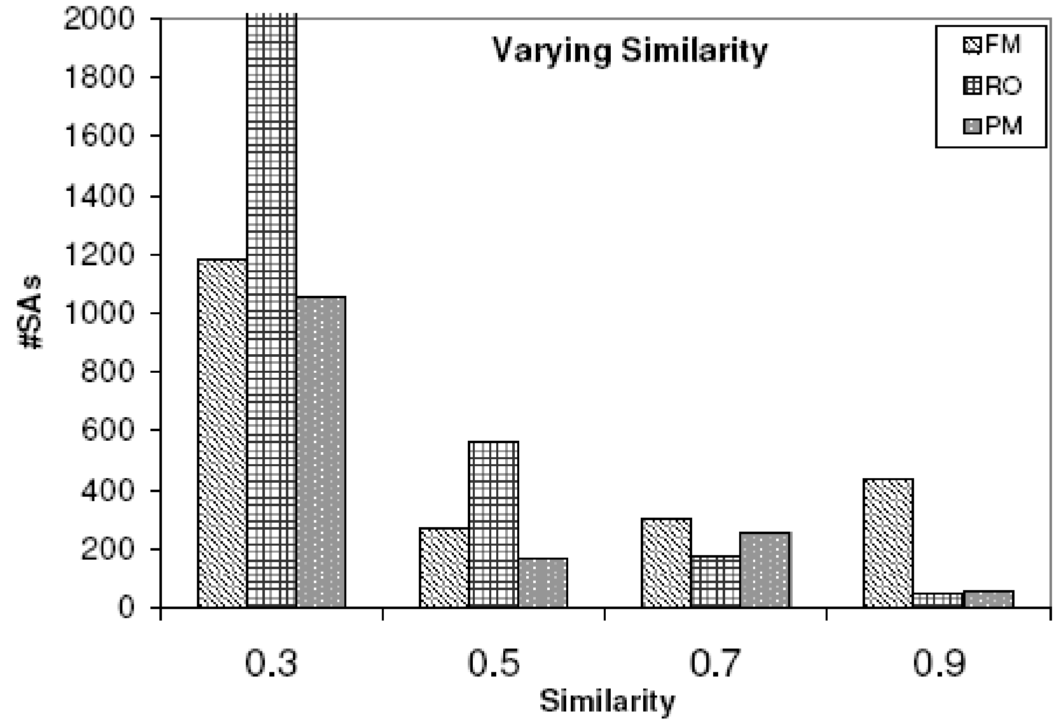
Quality Results (3)



RV80
RP80
RV20
RP20

Performance Results

- m=3
- Group Size=5
- 10 recommended items



Outline

- Recommendation for individuals
- Group recommendation
- Experiments
- Conclusion

Conclusion

- An approach that gives better results for large, dissimilar groups
- Tuning Problem (weights, m)
- PM: materializing in preprocessing time can be very time consuming (new ratings?)
- PM: depends on knowing the p(G) distribution
- Threshold Sharpening: nice, but a difficult optimization problem
- PM vs Threshold Sharpening

Thank you for your Attention

Questions?