Group Recommendation: Semantics and Efficiency

Sihem Amer-Yahia, Senjuti Basu-Roy, Ashish Chawla, Gautam Das, and Cong Yu.

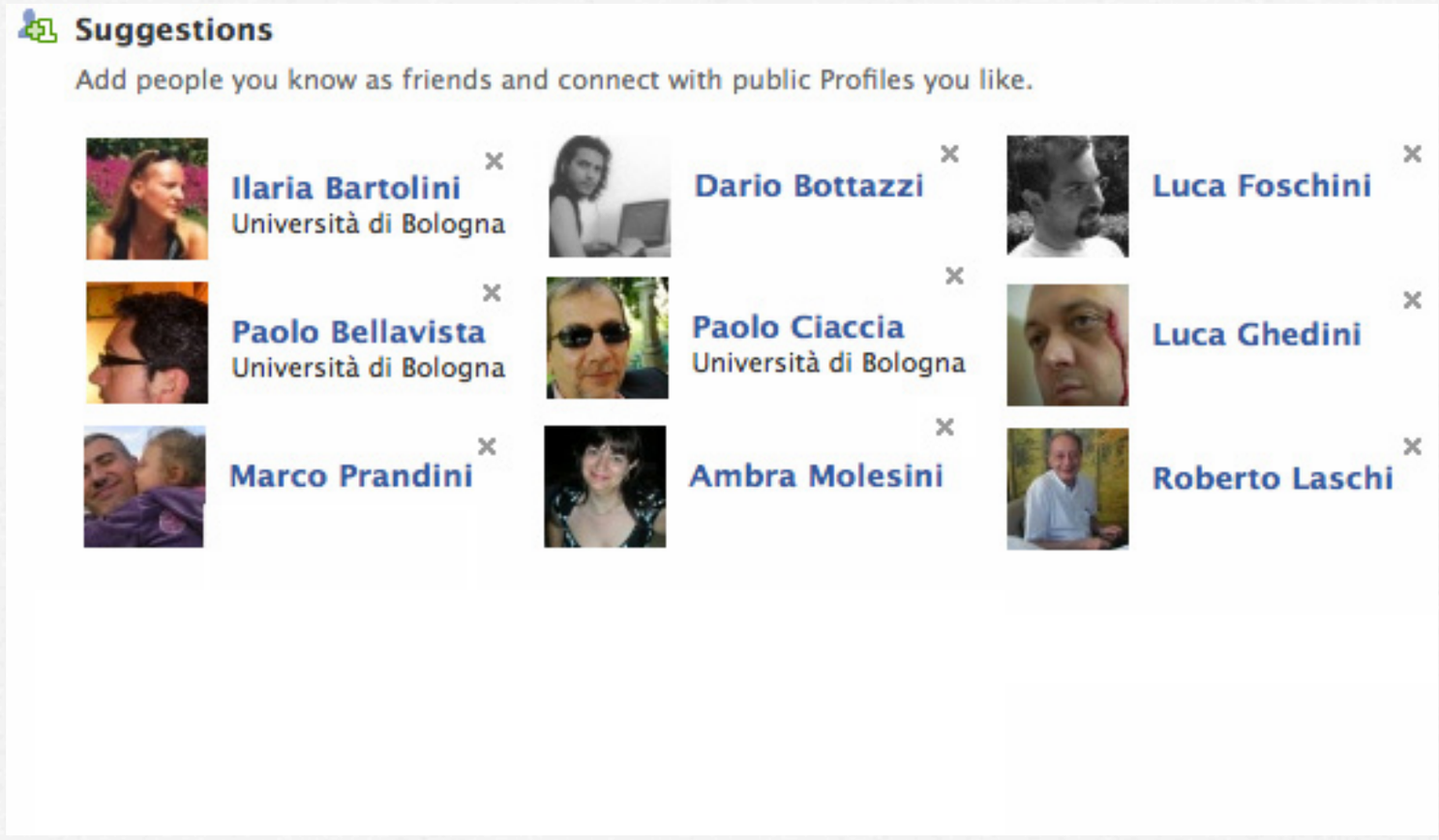
Gruppo 7:

Marco D'Alessandro (R)
Angelo Davide Cinardo
Pierpaolo Tommasi

Individual Recommendation

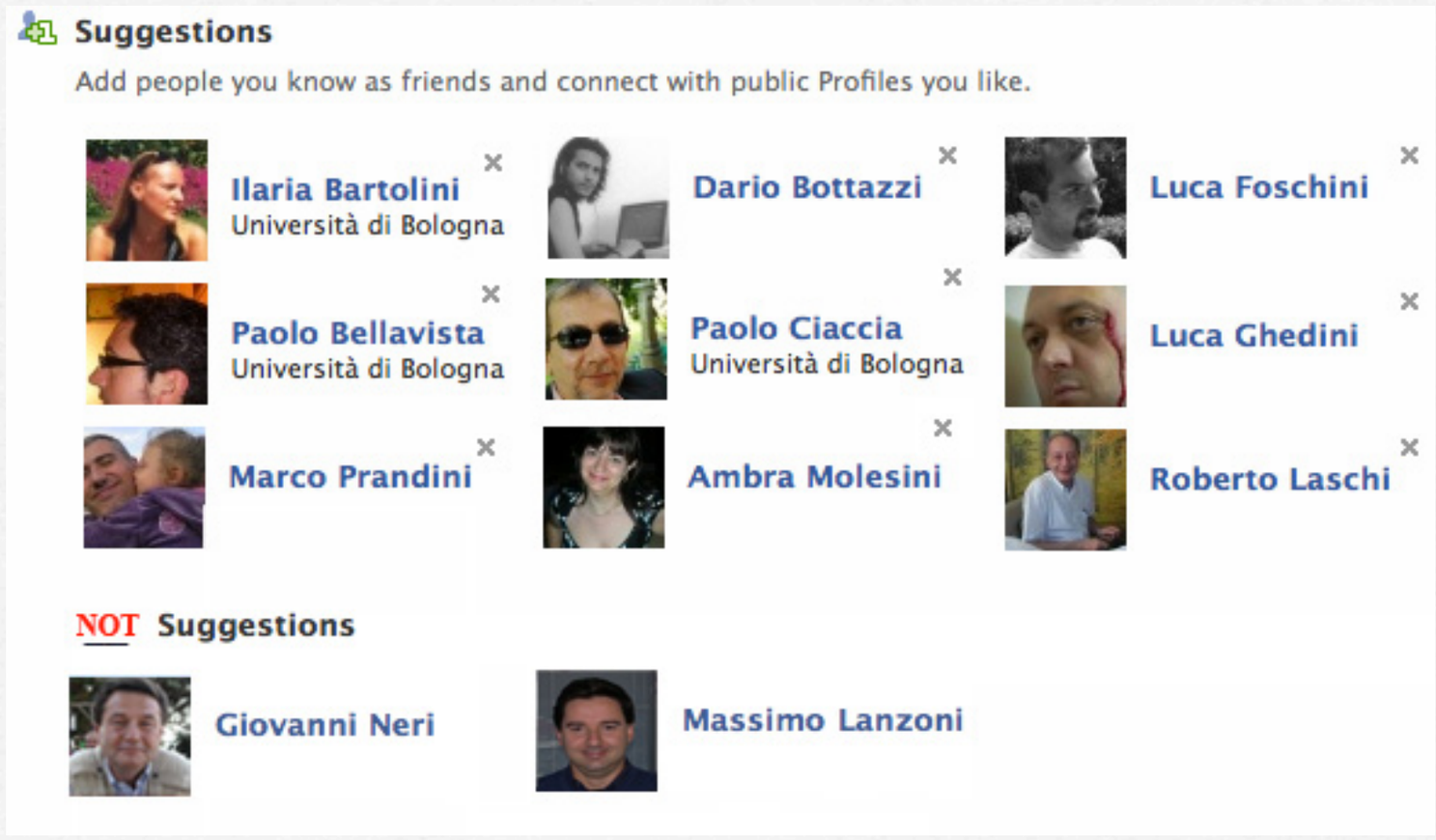
- [ ] Che libro dovrei leggere?

Individual Recommendation



- Quale amico dovrei aggiungere?

Individual Recommendation



- Quale amico non devo aggiungere?

Individual Recommendation

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

- Strategie di due tipi:
  - item-based: item simili già valutati
  - collaborative filtering: item valutati da utente simile

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

Individual Recommendation

Molti algoritmi e servizi di I.R. per soddisfare i nostri gusti..

Individual Recommendation

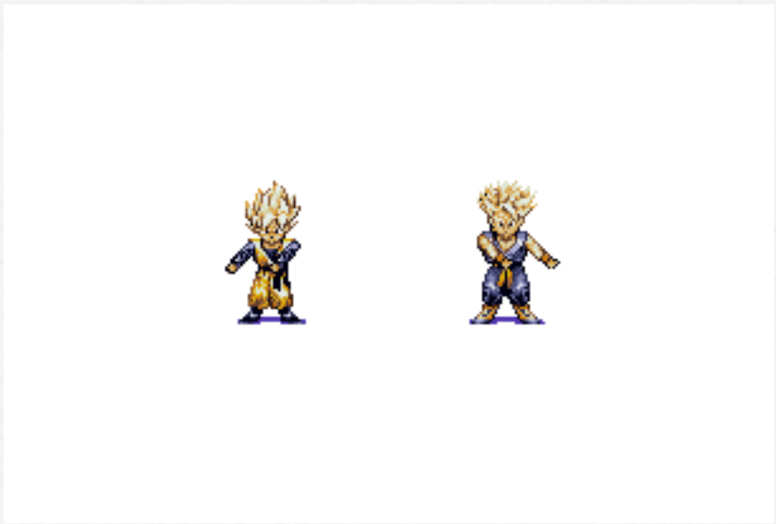
.. e se non siamo soli?

Group Recommendation

Aiutare un gruppo di conoscenti a trovare risorse che possano soddisfare i gusti generali

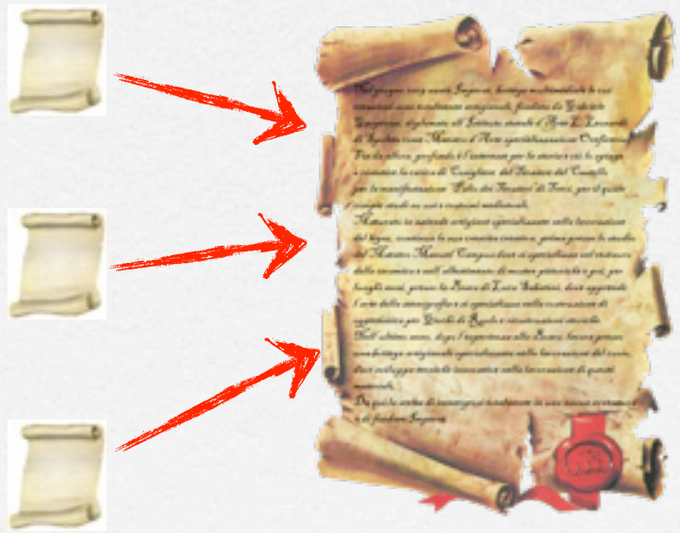
Group Recommendation

- Pochi lavori sui G.R.
- Soluzioni esistenti:
  - Preference Aggregation: Aggrega gli item più votati dai singoli, in un utente virtuale a cui fornire Recommendation.



Group Recommendation

- Pochi lavori sui G.R.
- Soluzioni esistenti:
  - Score Aggregation: aggrega I.R. dei singoli, per creare un'unica lista per il gruppo:
    - Average
    - Least Misery



Perché non bastano?

- Obiettivo: consigliare un film a un gruppo $G = \{u1, u2, u3\}$

- relevance (u1, "Amici miei") = 5
- relevance (u2, "Amici miei") = 1
- relevance (u3, "Amici miei") = 1

- relevance (u1, "Via col vento") = 3
- relevance (u2, "Via col vento") = 3
- relevance (u3, "Via col vento") = 1

Stesso score medio

↓

Average NO :(

Perché non bastano?

- Obiettivo: consigliare un film a un gruppo $G=\{u1, u2, u3\}$

- relevance (u1, "Amici miei") = 5
- relevance (u2, "Amici miei") = 1
- relevance (u3, "Amici miei") = 1

- relevance (u1, "Via col vento") = 3
- relevance (u2, "Via col vento") = 3
- relevance (u3, "Via col vento") = 1

Least misery NO :(

Group Disagreement

"..despite being friends, people may not share the same tastes for all things.."

- Un item può essere valutato diversamente da membri dello stesso gruppo
- sono parametri da considerare in Group Recommendation

Group Disagreement

Average Pair-wise

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

Disagreement Variance

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

Obiettivo: calcolare uno score per ogni item che rifletta gli interessi e le preferenze del gruppo

Ingredienti:

- Esprimere il gradimento di un singolo item per il gruppo
- Esprimere il livello di disaccordo tra i vari membri riguardo gli item

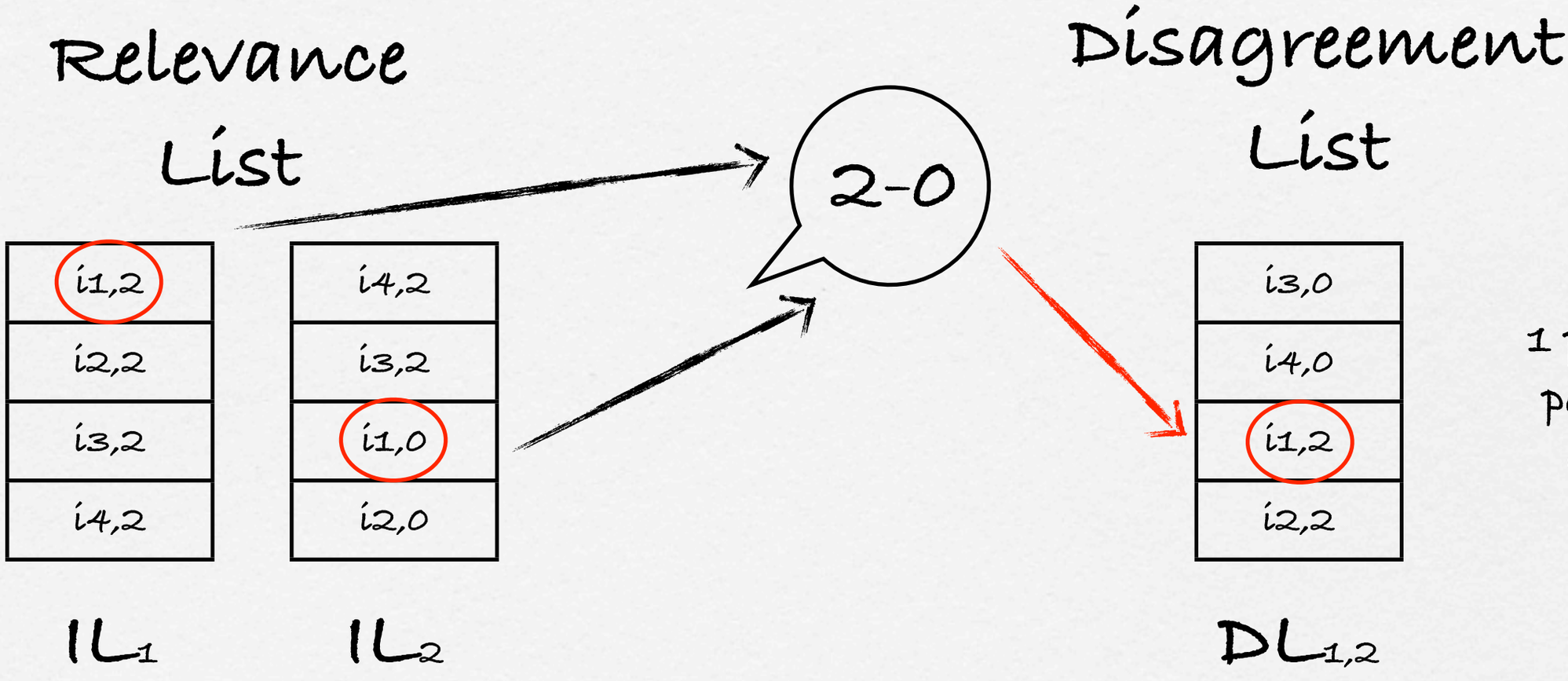
Ingredienti:

CONSENSUS FUNCTION

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

$$w_1 + w_2 = 1.0$$

Strutture di supporto:



1 per ogni possibile coppia

$$\Delta^{i}_{u,v} = |relevance(u,i) - relevance(v,i)|$$

Definizione del problema

Dato un gruppo G e una Consensus Function F, si deve ritornare una lista di K elementi, in ordine decrescente secondo i valori di F, tale che ogni elemento sia nuovo per tutti i membri del gruppo

↕

Top-K Group Recommendation

Threshold Algorithm

I migliori algoritmi per risolvere problemi di tipo top-K appartengono alla famiglia dei T.A.

Per garantire efficienza si possono applicare condizioni di STOP basate su score bounds.

monotonicità

DL monotone

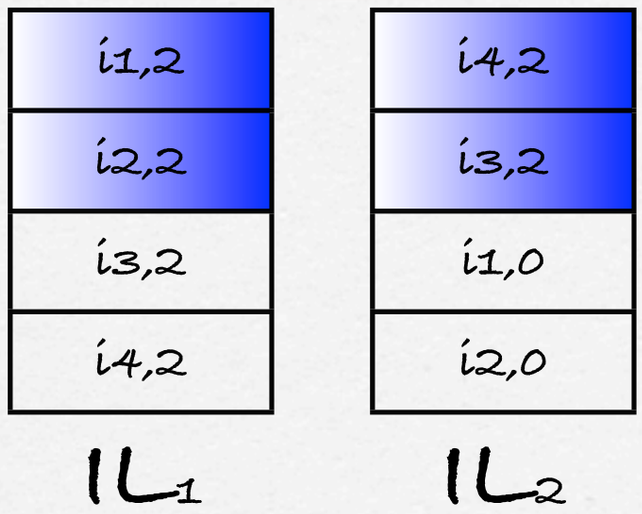
iff

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

- Average
- Least-misery
- Pair-wise
- Variance

Efficient Recommendation Computation

4 Sorted Access per top-1: i3



- Ordina in maniera decrescente IL
- Applica T.A.

Si può migliorare?

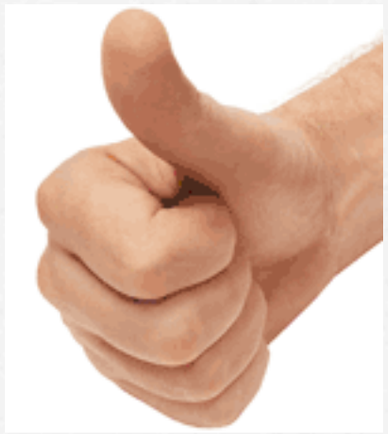
Pruning grazie alle Disagreement List

Efficient Recommendation Computation

4 Sorted Access per top-1: i3

IL_1
i1,2
i2,2
i3,2
i4,2

IL_2
i4,2
i3,2
i1,0
i2,0



- Ordina in maniera decrescente IL
- Applica T.A.

3 Sorted Access per top-1: i3

$DL_{1,2}$
i3,0
i4,0
i1,2
i2,2

- Ordina in maniera crescente pair-wise DL

Si può migliorare?

Pruning grazie alle Disagreement List

Group Recommendation Algorithm

Group G

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

IL1	IL2	DL1,2
i1,2	i4,2	i3,0
i2,2	i3,2	i4,0
i3,2	i1,0	i1,2
i4,2	i2,0	i2,2

Algorithm 1 Group Recommendation Algorithm with Fully Materialized Disagreement Lists (FM)

Require: Group $\mathcal{G}$, consensus function F;
1: Retrieve relevance lists $\mathcal{IL}_u$ for each user u in group $\mathcal{G}$;
2: Retrieve disagreement lists $\mathcal{DL}_{(u,v)}$ for each user pair (u, v) in group $\mathcal{G}$;
3: Cursor cur = **getNext**() moves across each relevance and disagreement lists;
4: **while** (cur <> NULL) **do**
5: Get entry $e = (i, r)$ at cur;
6: **if** not(**inHeap**(**topKHeap**, e)) **then**
7: **if** (**ComputeMaxScore**($e.i, e.r$, F) $\geq$ **topKHeap**.$k_{th}score$) **then**
8: **ComputeExactScore**; Probe $\mathcal{IL}s$ to compute exact score $score$ of e using F;
9: **topKHeap.addToHeap**($e.i, score$);
10: **end if**
11: **end if**
12: cur = **getNext**();
13: **end while**
14: **return** topKList(**topKHeap**);

Group Recommendation Algorithm

Fully Materialized DL (FM):

- SA su ogni input list (Round Robin) tramite getNext()
- ComputeExactScore(): esegue Random Access su IL per definire lo score di ogni elemento (usando F).
- ComputeMaxScore(): produce un nuovo Threshold ad ogni iterazione (usando F) UpperBound

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

Algorithm 1 Group Recommendation Algorithm with Fully Materialized Disagreement Lists (FM)

Require: Group $\mathcal{G}$, consensus function F;
1: Retrieve relevance lists $\mathcal{IL}_u$ for each user u in group $\mathcal{G}$;
2: Retrieve disagreement lists $\mathcal{DL}_{(u,v)}$ for each user pair (u, v) in group $\mathcal{G}$;
3: Cursor *cur* = **getNext()** moves across each relevance and disagreement lists;
4: **while** (*cur* <> NULL) **do**
5: Get entry $e = (i, r)$ at *cur*;
6: **if** not(**inHeap**(**topKHeap**, e)) **then**
7: **if** (**ComputeMaxScore**($e.i, e.r$, F) $\geq$ **topKHeap**.$k_{th}score$) **then**
8: **ComputeExactScore**; Probe $\mathcal{IL}s$ to compute exact score *score* of e using F;
9: **topKHeap.addToHeap**($e.i$, *score*);
10: **end if**
11: **end if**
12: *cur* = **getNext**();
13: **end while**
14: **return** topKList(**topKHeap**);

Group Recommendation Algorithm

Fully Materialized DL (FM):

- SA su ogni input list (Round Robin) tramite getNext()
- ComputeExactScore(): esegue Random Access su IL per definire lo score di ogni elemento (usando F).
- ComputeMaxScore(): produce un nuovo Threshold ad ogni iterazione (usando F) UpperBound

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

Algorithm 1 Group Recommendation Algorithm with Fully Materialized Disagreement Lists (FM)

Require: Group $\mathcal{G}$, consensus function F;
1: Retrieve relevance lists $\mathcal{IL}_u$ for each user u in group $\mathcal{G}$;
2: Retrieve disagreement lists $\mathcal{DL}_{(u,v)}$ for each user pair (u, v) in group $\mathcal{G}$;
3: Cursor cur = **getNext()** moves across each relevance and disagreement lists;
4: **while** (cur <> NULL) **do**
5: Get entry $e = (i, r)$ at cur;
6: **if** not(**inHeap**(**topKHeap**, e)) **then**
7: **if** (**ComputeMaxScore**($e.i, e.r$, F) $\geq$ **topKHeap**.$k_{th}score$) **then**
8: **ComputeExactScore**; Probe $\mathcal{IL}s$ to compute exact score $score$ of e using F;
9: **topKHeap.addToHeap**($e.i, score$);
10: **end if**
11: **end if**
12: cur = **getNext**();
13: **end while**
14: **return** topKList(**topKHeap**);

Relevance List Only (RO):

- Non utilizza le Disagreement List

Group Recommendation Algorithm

Fully Materialized DL (FM):

- SA su ogni input list (Round Robin) tramite getNext()

- ComputeExactScore(): esegue Random Access su IL per definire lo score di ogni elemento (usando F).

- ComputeMaxScore(): produce un nuovo Threshold ad ogni iterazione (usando F) UpperBound

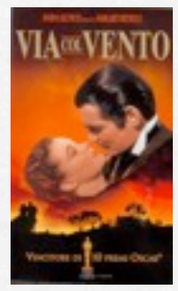
- Migliore efficienza
- Numero di DL elevato

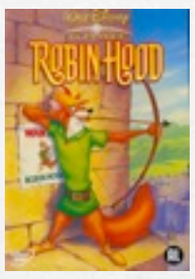
Relevance List Only (RO):

- Non utilizza le Disagreement List

- Upper Bound meno efficaci
- Minore occupazione di memoria

Esempio - RO





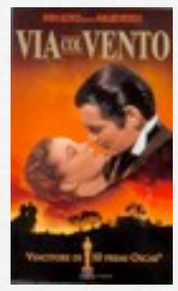
- Obiettivo: Elemento Top-1 su un gruppo di 3 utenti $G=\{u1, u2, u3\}$
- Input: 3 Relevance List (IL_1, IL_2, IL_3) su 4 film
- Performance: numero di SA

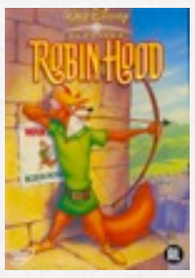
IL_1
i1,4
i3,4
i4,4
i2,2

IL_2
i2,4
i4,4
i1,2
i3,2

IL_3
i3,3
i1,3
i4,3
i2,3

Esempio - RO





- Obiettivo: Elemento Top-1 su un gruppo di 3 utenti G={u1, u2, u3}
- Input: 3 Relevance List (IL1, IL2, IL3) su 4 film
- Performance: numero di SA

IL1
i1,4
i3,4
i4,4
i2,2

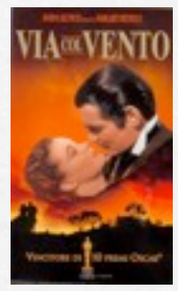
IL2
i2,4
i4,4
i1,2
i3,2

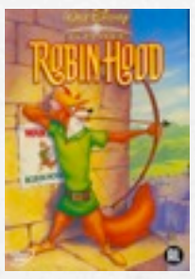
IL3
i3,3
i1,3
i4,3
i2,3

Top-K Buffer
i1,1.33

Th=1,93

Esempio - RO





- Obiettivo: Elemento Top-1 su un gruppo di 3 utenti G={u1, u2, u3}
- Input: 3 Relevance List (IL1, IL2, IL3) su 4 film
- Performance: numero di SA

IL1
i1,4
i3,4
i4,4
i2,2

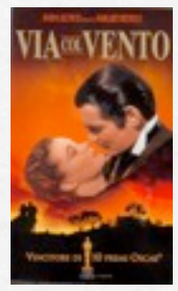
IL2
i2,4
i4,4
i1,2
i3,2

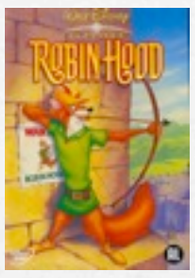
IL3
i3,3
i1,3
i4,3
i2,3

Top-K Buffer
i1,1.33
i2,1.33

Th=1,86

Esempio - RO





- Obiettivo: Elemento Top-1 su un gruppo di 3 utenti G={u1, u2, u3}
- Input: 3 Relevance List (IL1, IL2, IL3) su 4 film
- Performance: numero di SA

IL1
i1,4
i3,4
i4,4
i2,2

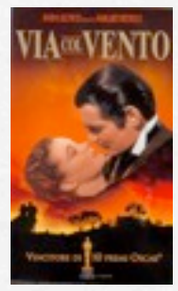
IL2
i2,4
i4,4
i1,2
i3,2

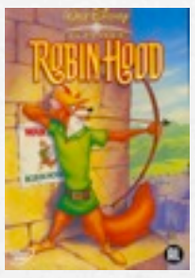
IL3
i3,3
i1,3
i4,3
i2,3

Top-K Buffer
i1 1,33
i2,1.33
i3,1.33

Th=1,73

Esempio - RO





- Obiettivo: Elemento Top-1 su un gruppo di 3 utenti G={u1, u2, u3}
- Input: 3 Relevance List (IL1, IL2, IL3) su 4 film
- Performance: numero di SA

IL1
i1,4
i3,4
i4,4
i2,2

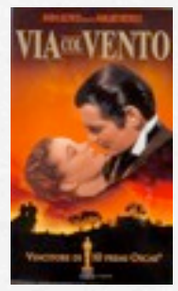
IL2
i2,4
i4,4
i1,2
i3,2

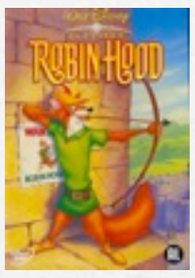
IL3
i3,3
i1,3
i4,3
i2,3

Top-K Buffer
i1 1,33
i2,1.33
i3,1.33

Th=1,73

Esempio - RO





- Obiettivo: Elemento Top-1 su un gruppo di 3 utenti G={u1, u2, u3}
- Input: 3 Relevance List (IL1, IL2, IL3) su 4 film
- Performance: numero di SA

IL1
i1,4
i3,4
i4,4
i2,2

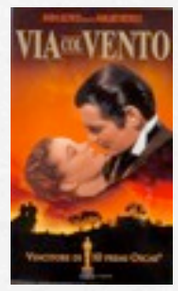
IL2
i2,4
i4,4
i1,2
i3,2

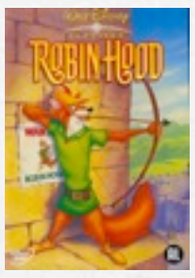
IL3
i3,3
i1,3
i4,3
i2,3

Top-K Buffer
i4,1.6
i1 1,33
i2,1.33
i3,1.33

Th=1,73

Esempio - RO





- Obiettivo: Elemento Top-1 su un gruppo di 3 utenti G={u1, u2, u3}
- Input: 3 Relevance List (IL1, IL2, IL3) su 4 film
- Performance: numero di SA

IL1
i1,4
i3,4
i4,4
i2,2

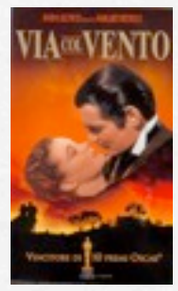
IL2
i2,4
i4,4
i1,2
i3,2

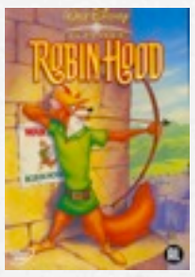
IL3
i3,3
i1,3
i4,3
i2,3

Top-K Buffer
i4,1.6
i1 1,33
i2,1.33
i3,1.33

Th=1,73

Esempio - RO





- Obiettivo: Elemento Top-1 su un gruppo di 3 utenti G={u1, u2, u3}
- Input: 3 Relevance List (IL1, IL2, IL3) su 4 film
- Performance: numero di SA

IL1
i1,4
i3,4
i4,4
i2,2

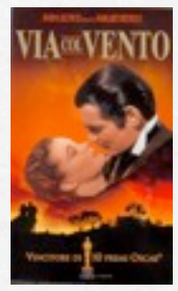
IL2
i2,4
i4,4
i1,2
i3,2

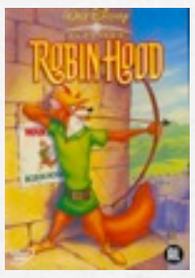
IL3
i3,3
i1,3
i4,3
i2,3

Top-K Buffer
i4,1.6
i1 1,33
i2,1.33
i3,1.33

Th=1,73

Esempio - RO





- Obiettivo: Elemento Top-1 su un gruppo di 3 utenti G={u1, u2, u3}
- Input: 3 Relevance List (IL1, IL2, IL3) su 4 film
- Performance: numero di SA

IL1
i1,4
i3,4
i4,4
i2,2

IL2
i2,4
i4,4
i1,2
i3,2

IL3
i3,3
i1,3
i4,3
i2,3

Top-K Buffer
i4,1.6
i1 1,33
i2,1.33
i3,1.33

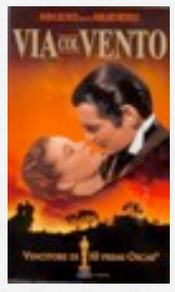
Th=1,60

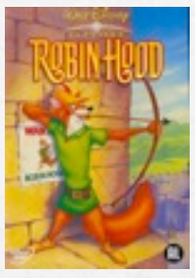
STOP!

Top-1 = i4

#SA=8

Esempio - FM





- Input: 3 Relevance List (IL1, IL2, IL3) e 3 pair-wise Disagreement List (DL12, DL13, DL23)

IL1
i1,4
i3,4
i4,4
i2,2

IL2
i2,4
i4,4
i1,2
i3,2

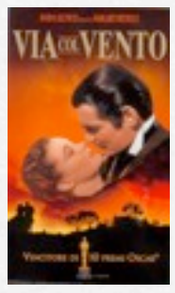
IL3
i3,3
i1,3
i4,3
i2,3

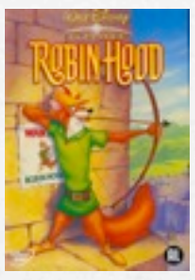
DL12
i4,0
i3,2
i2,2
i1,2

DL13
i4,1
i3,1
i2,1
i1,1

DL23
i4,1
i1,1
i3,1
i2,1

Esempio - FM





- Input: 3 Relevance List (IL1, IL2, IL3) e 3 pair-wise Disagreement List (DL12, DL13, DL23)

IL1
i1,4
i3,4
i4,4
i2,2

IL2
i2,4
i4,4
i1,2
i3,2

IL3
i3,3
i1,3
i4,3
i2,3

DL12
i4,0
i3,2
i2,2
i1,2

DL13
i4,1
i3,1
i2,1
i1,1

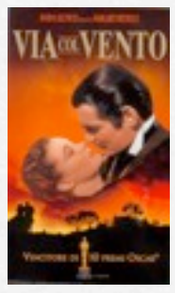
DL23
i4,1
i1,1
i3,1
i2,1

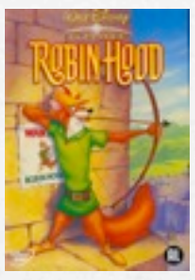
Top-K buffer

i1,1.33

Th=1,93

Esempio - FM





- Input: 3 Relevance List (IL1, IL2, IL3) e 3 pair-wise Disagreement List (DL12, DL13, DL23)

IL1
i1,4
i3,4
i4,4
i2,2

IL2
i2,4
i4,4
i1,2
i3,2

IL3
i3,3
i1,3
i4,3
i2,3

DL12
i4,0
i3,2
i2,2
i1,2

DL13
i4,1
i3,1
i2,1
i1,1

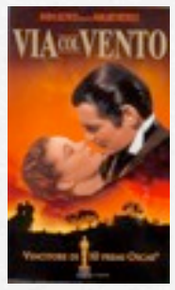
DL23
i4,1
i1,1
i3,1
i2,1

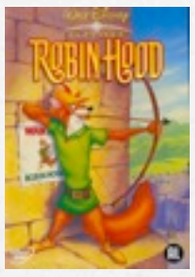
Top-K buffer

i1 1,33
i2,1.33

Th=1,86

Esempio - FM





- Input: 3 Relevance List (IL1, IL2, IL3) e 3 pair-wise Disagreement List (DL12, DL13, DL23)

IL1
i1,4
i3,4
i4,4
i2,2

IL2
i2,4
i4,4
i1,2
i3,2

IL3
i3,3
i1,3
i4,3
i2,3

DL12
i4,0
i3,2
i2,2
i1,2

DL13
i4,1
i3,1
i2,1
i1,1

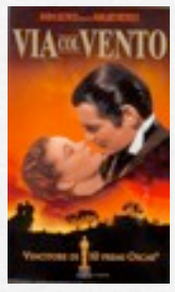
DL23
i4,1
i1,1
i3,1
i2,1

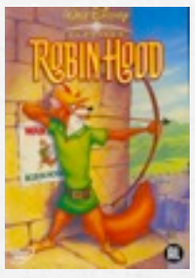
Top-K buffer

i1 1,33
i2,1.33
i3,1.33

Th=1,73

Esempio - FM





- Input: 3 Relevance List (IL1, IL2, IL3) e 3 pair-wise Disagreement List (DL12, DL13, DL23)

IL1
i1,4
i3,4
i4,4
i2,2

IL2
i2,4
i4,4
i1,2
i3,2

IL3
i3,3
i1,3
i4,3
i2,3

DL12
i4,0
i3,2
i2,2
i1,2

DL13
i4,1
i3,1
i2,1
i1,1

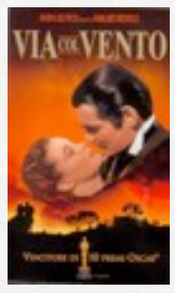
DL23
i4,1
i1,1
i3,1
i2,1

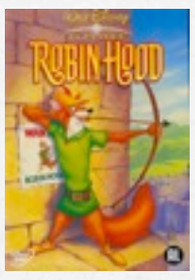
Top-K buffer

14,1.6
i1 1,33
i2,1.33
i3,1.33

Th=1,73

Esempio - FM





- Input: 3 Relevance List (IL1, IL2, IL3) e 3 pair-wise Disagreement List (DL12, DL13, DL23)

IL1
i1,4
i3,4
i4,4
i2,2

IL2
i2,4
i4,4
i1,2
i3,2

IL3
i3,3
i1,3
i4,3
i2,3

DL12
i4,0
i3,2
i2,2
i1,2

DL13
i4,1
i3,1
i2,1
i1,1

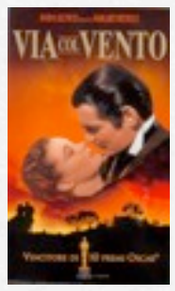
DL23
i4,1
i1,1
i3,1
i2,1

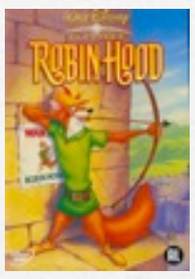
Top-K buffer

14,1.6
i1 1,33
i2,1.33
i3,1.33

Th=1,66

Esempio - FM





- Input: 3 Relevance List (IL1, IL2, IL3) e 3 pair-wise Disagreement List (DL12, DL13, DL23)

IL1
i1,4
i3,4
i4,4
i2,2

IL2
i2,4
i4,4
i1,2
i3,2

IL3
i3,3
i1,3
i4,3
i2,3

DL12
i4,0
i3,2
i2,2
i1,2

DL13
i4,1
i3,1
i2,1
i1,1

DL23
i4,1
i1,1
i3,1
i2,1

Top-K buffer

14,1.6
i1 1,33
i2,1.33
i3,1.33

Th=1,60

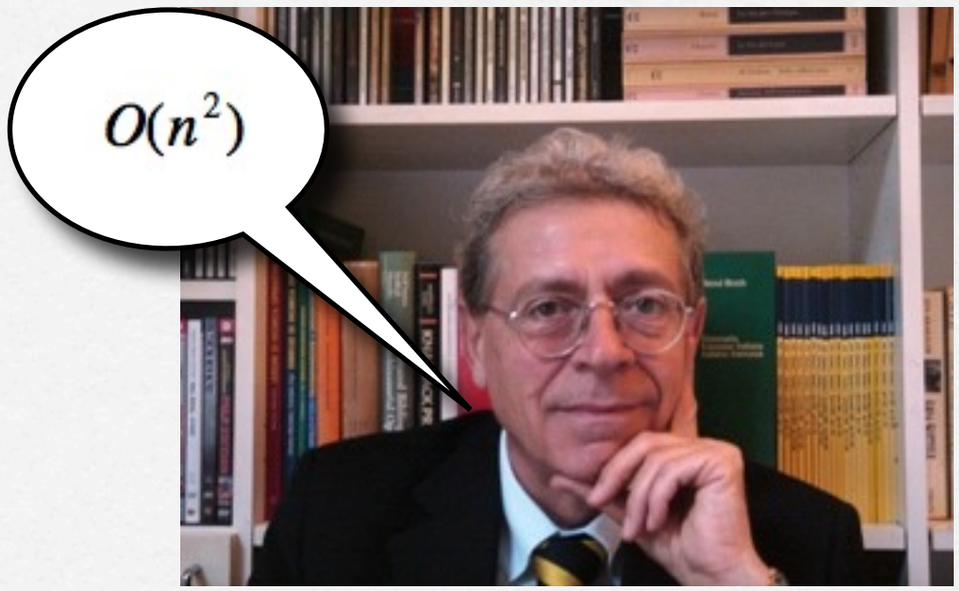
STOP!

Top-1 = i4

#SA=6

Ottimizzazioni

- Dati n utenti il numero di pair-wise Disagreement List è $\frac{n(n-1)}{2}$



Ottimizzazioni

- Dati n utenti il numero di pair-wise Disagreement List è $\frac{n(n-1)}{2}$

Partial Materialization:
Materializzare solo un sottoinsieme m di DL in preprocessing-step.

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

Partial Materialization

- Dati 5.000 utenti ⟶ #DL = 12.497.500
- Elevata memoria richiesta

<u>Intuizione</u>: materializzare DL se:

1) È molto probabile che gli utenti si riuniscano in un gruppo e richiedano Recommendation

2) la loro presenza aumenta l'efficienza delle query Top-K.

Partial Materialization

1) P(Gruppo)
2) Efficienza

Soluzione

coppia di utenti	#SA senza DL	#SA con DL	Differenza
{u1,u2}	200	100	100
{u3,u4}	290	195	95
{u10,u9}	170	100	70
{u6,u7}	230	190	40
{u2,u3}	175	145	30
…	…	…	…

m

$O(n^2)$

$$p(\{u,v\}) = \frac{|\{\mathcal{G}_i | u,v \in \mathcal{G}_i\}|}{r}$$

rich query log

Sharpening Threshold

Possiamo sfruttare le dipendenze tra relevance e disagreement list per migliorare il Threshold ???

ILu
i1,0.5
i3,0.5
…

ILv
i2,0.5
i3,0.4
…

DLuv
i3,0.2
i1,0.3
…

$i_u = 0.5$ $i_v = 0.5$

$$\mathrm{F}(\mathcal{G}, i) = (i_u + i_v)/2 + (1 - |i_u i_v|/1)$$

$$0 \leq i_u \leq 0.5$$
$$0 \leq i_v \leq 0.5$$
$$0.2 \leq |i_u - i_v| \leq 1$$

Th = 1.3

Sharpening Threshold

Possiamo sfruttare le dipendenze tra relevance e disagreement list per migliorare il Threshold ???

lLu
i1,0.5
i3,0.5
…

lLv
i2,0.5
i3,0.4
…

DLuv
i3,0.2
i1,0.3
…

$i_u = 0.5$ $i_v = 0.3$

$$\mathrm{F}(\mathcal{G}, i) = (i_u + i_v)/2 + (1 - |i_u i_v|/1)$$

~~Th = 1.3~~

$$0 \leq i_u \leq 0.5$$
$$0 \leq i_v \leq 0.5$$
$$0.2 \leq |i_u - i_v| \leq 1$$

Sharpening Threshold

Possiamo sfruttare le dipendenze tra relevance e disagreement list per migliorare il Threshold ???

ILu
i1,0.5
i3,0.5
…

ILv
i2,0.5
i3,0.4
…

DLuv
i3,0.2
i1,0.3
…

$$\mathbf{F}(\mathcal{G}, i) = (i_u + i_v)/2 + (1 - |i_u i_v|/1)$$

$$0 \leq i_u \leq 0.5$$
$$0 \leq i_v \leq 0.5$$
$$0.2 \leq |i_u - i_v| \leq 1$$

Th = 1.2

Prove Sperimentali

Il sito dà la possibilità di coinvolgere direttamente un gruppo di utenti nei test proposti dagli autori che ricevono dei feedback sulla bontà delle raccomandazioni fatte agli utenti.

- DataSet utilizzato: MovieLens

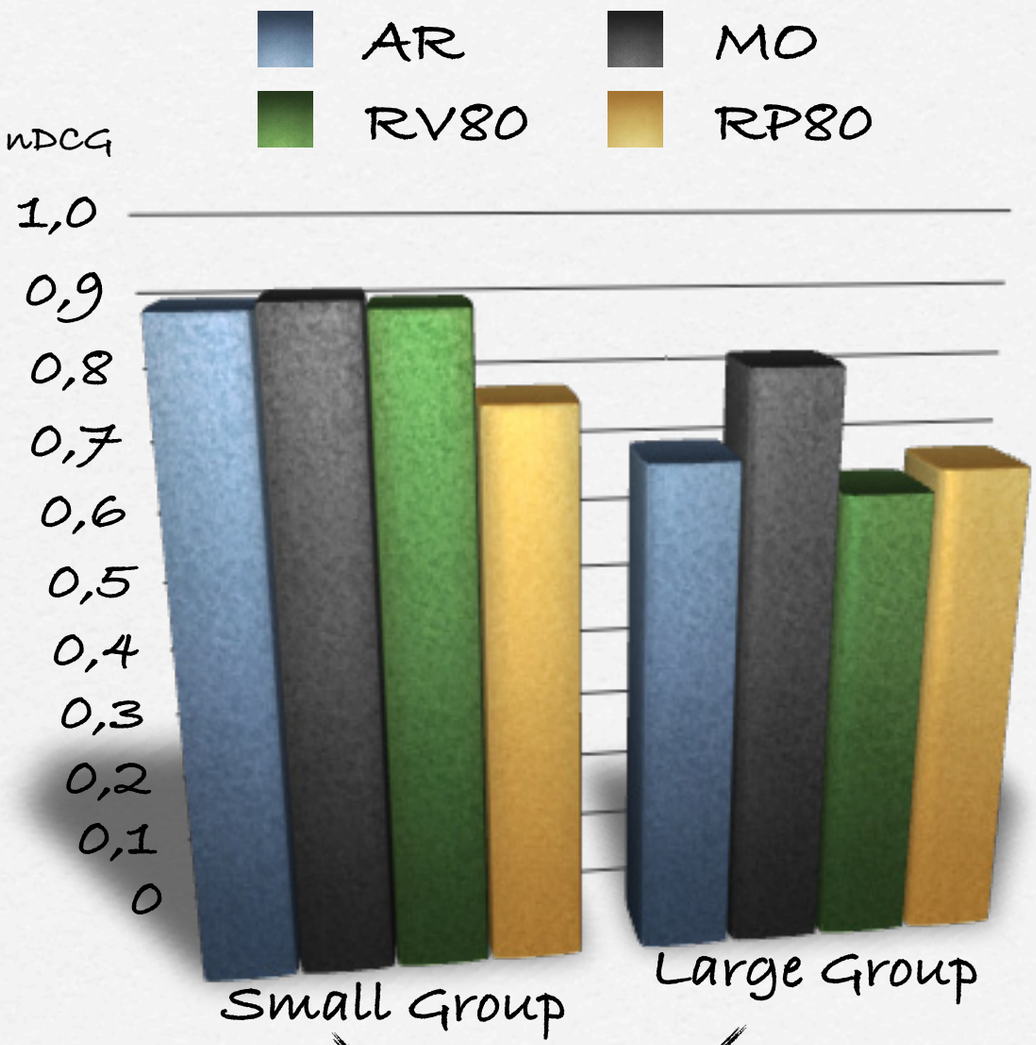
#utenti	#film	#voti
71.567	10.681	10.000.054

- Individual Recommendation calcolate con "collaborative filtering"
- Algoritmi di Group Recommendation testati sul set di utenti di amazonmechanical turk Artificial Artificial Intelligence
- 3 tipi di gruppi (similar, dissimilar e random) per 2 possibili dimensioni (da 3 e 8 utenti)

	Small Group	Large Group
Similar Group	0.89	0.90
Dissimilar Group	0.29	0.27
Random Group	0.69	0.73

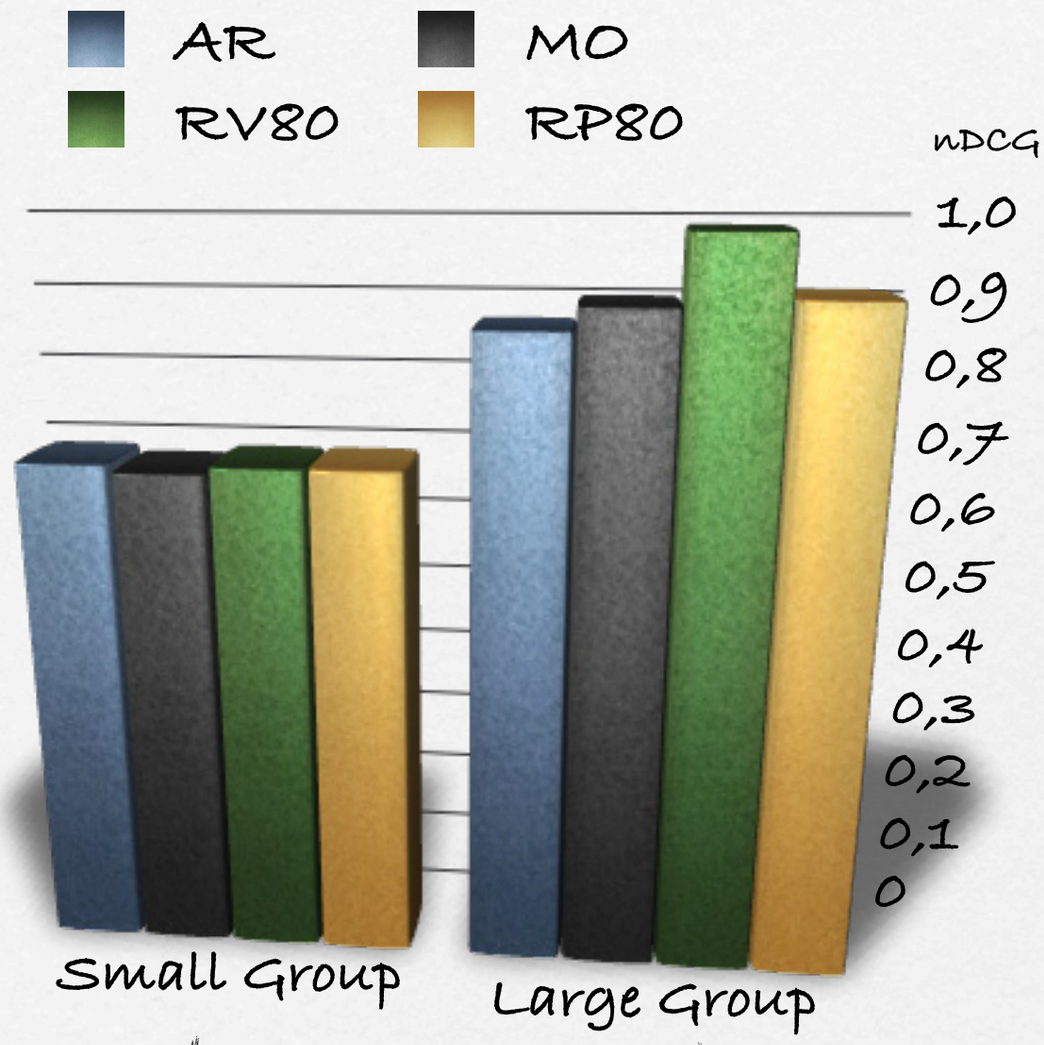
- 4 tipi di algoritmi testati: Average Relevance Only (AR), Least Misery Relevance Only (LM), Consensus with Pair-wise Disagreements (RP), Consensus with Disagreement Variance (RV)
- Valutata Performance su #SA comparando FM, RO e PM variando la dimensione e il tipo del gruppo e K
- Valutata efficacia della Partial Materialization
- Valutata efficacia del Threshold Sharpening

Similar User Group



Best: MO

Dissimilar User Group

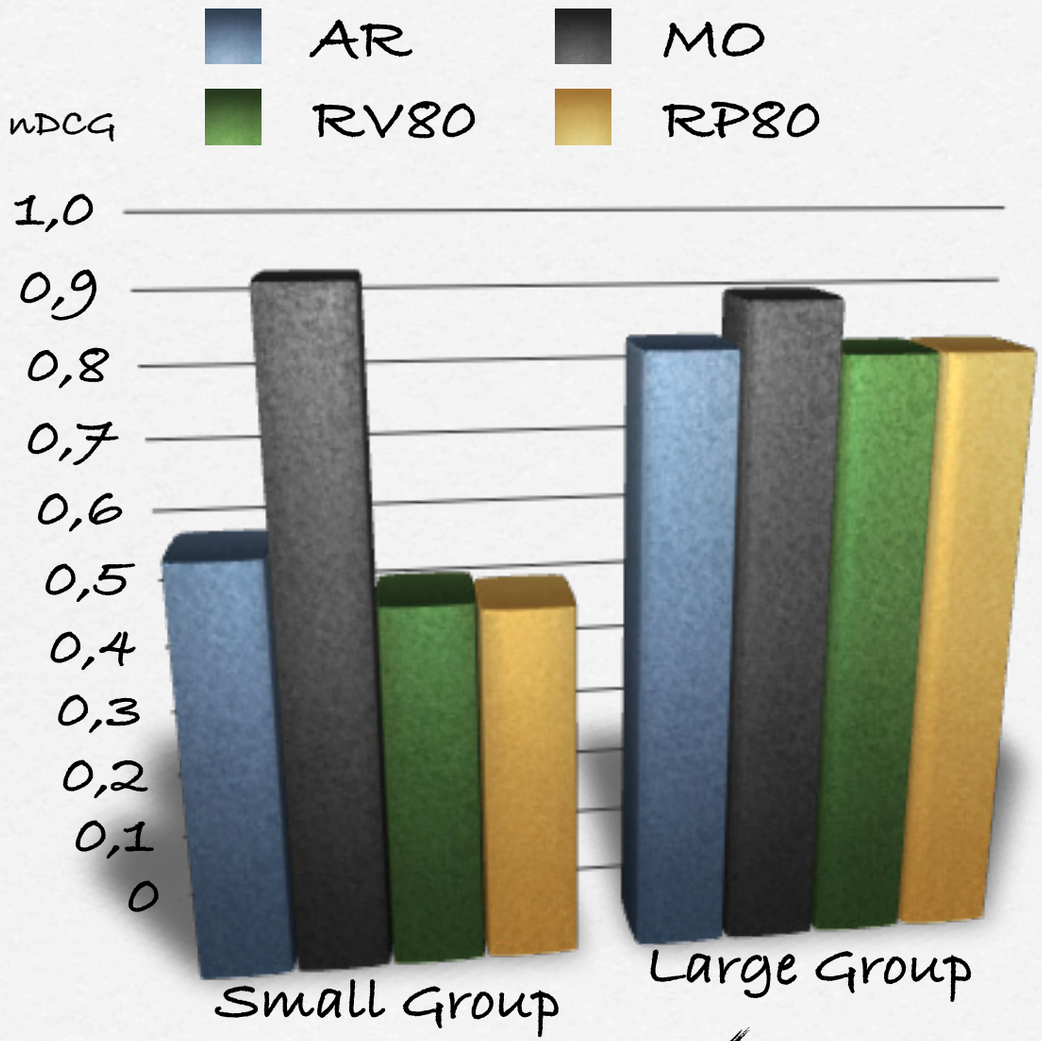


Best: RV80 RP80

Best: RV80

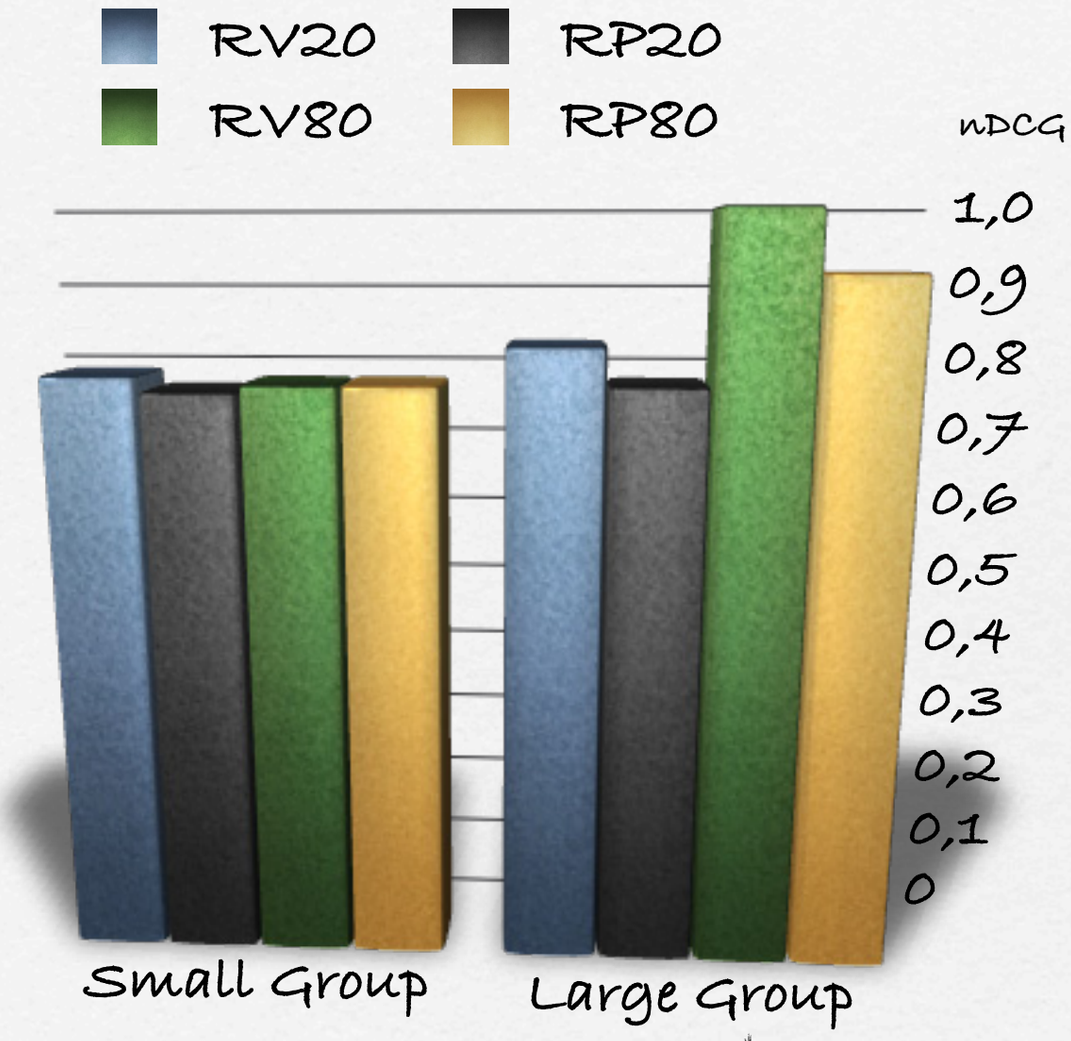
DCG = discounted cumulative gain: più algoritmo efficiente, più valore vicino ad 1

Random User Group



Best: MO

Dissimilar Group with variations

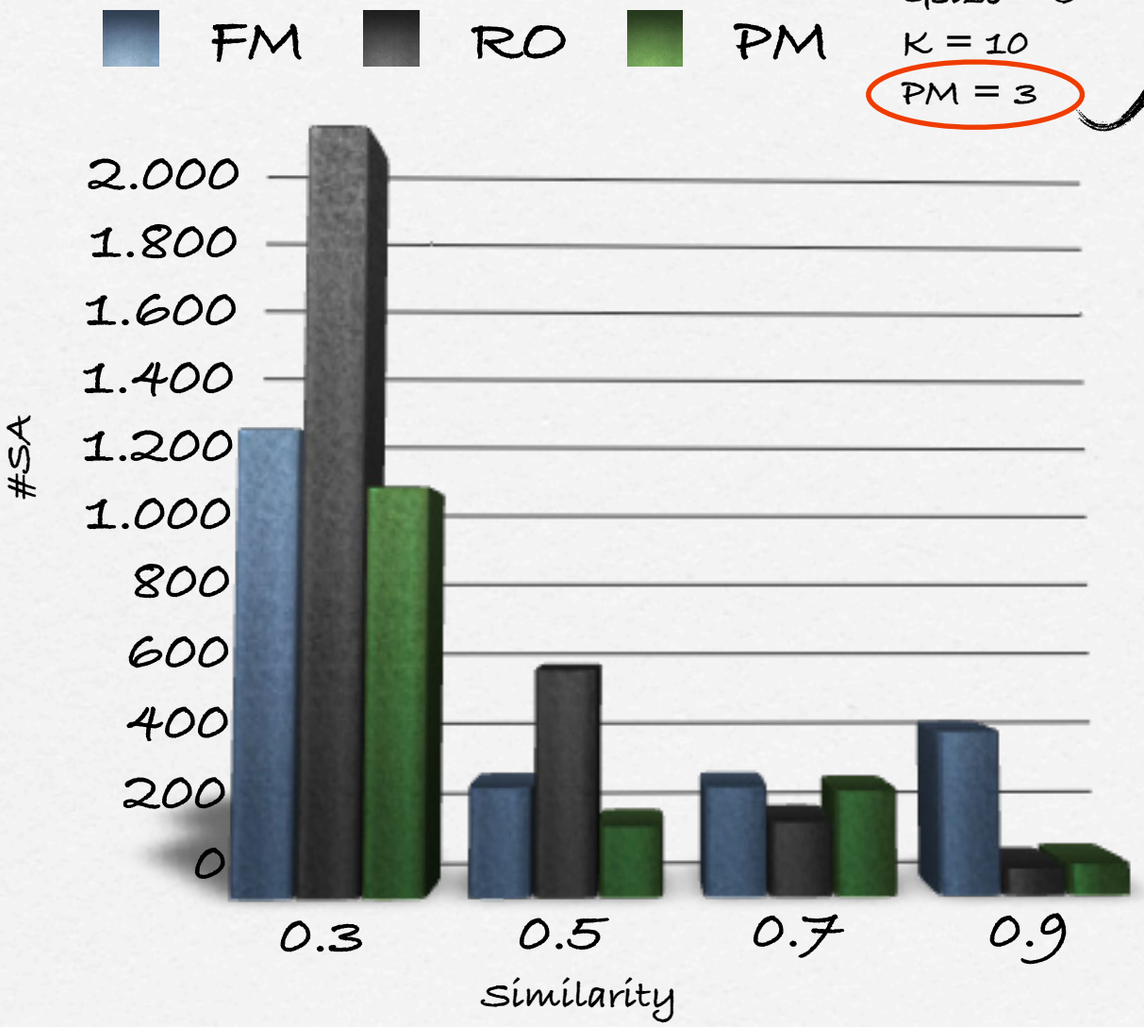


No Best

Best: RV80

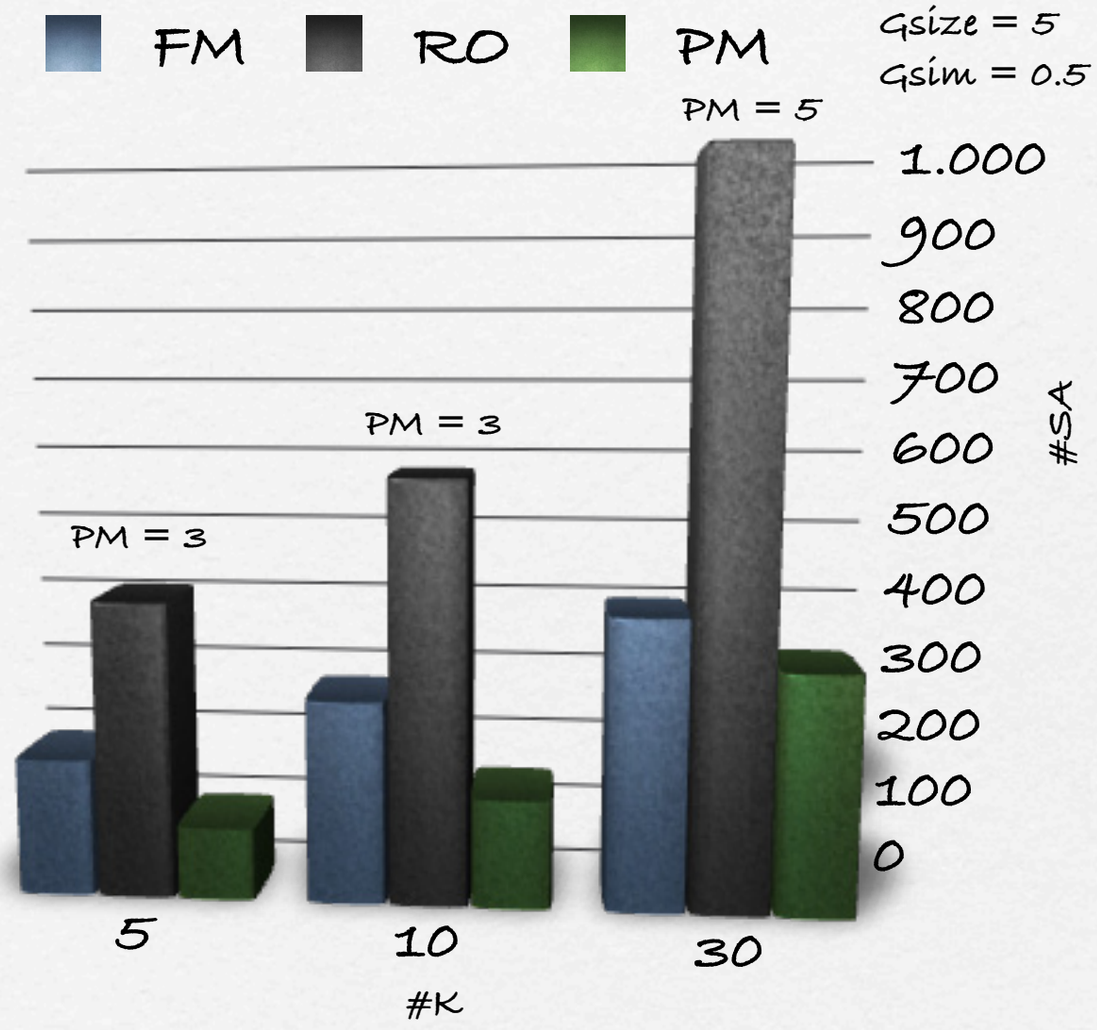
Varying Similarity

#Liste materializzate



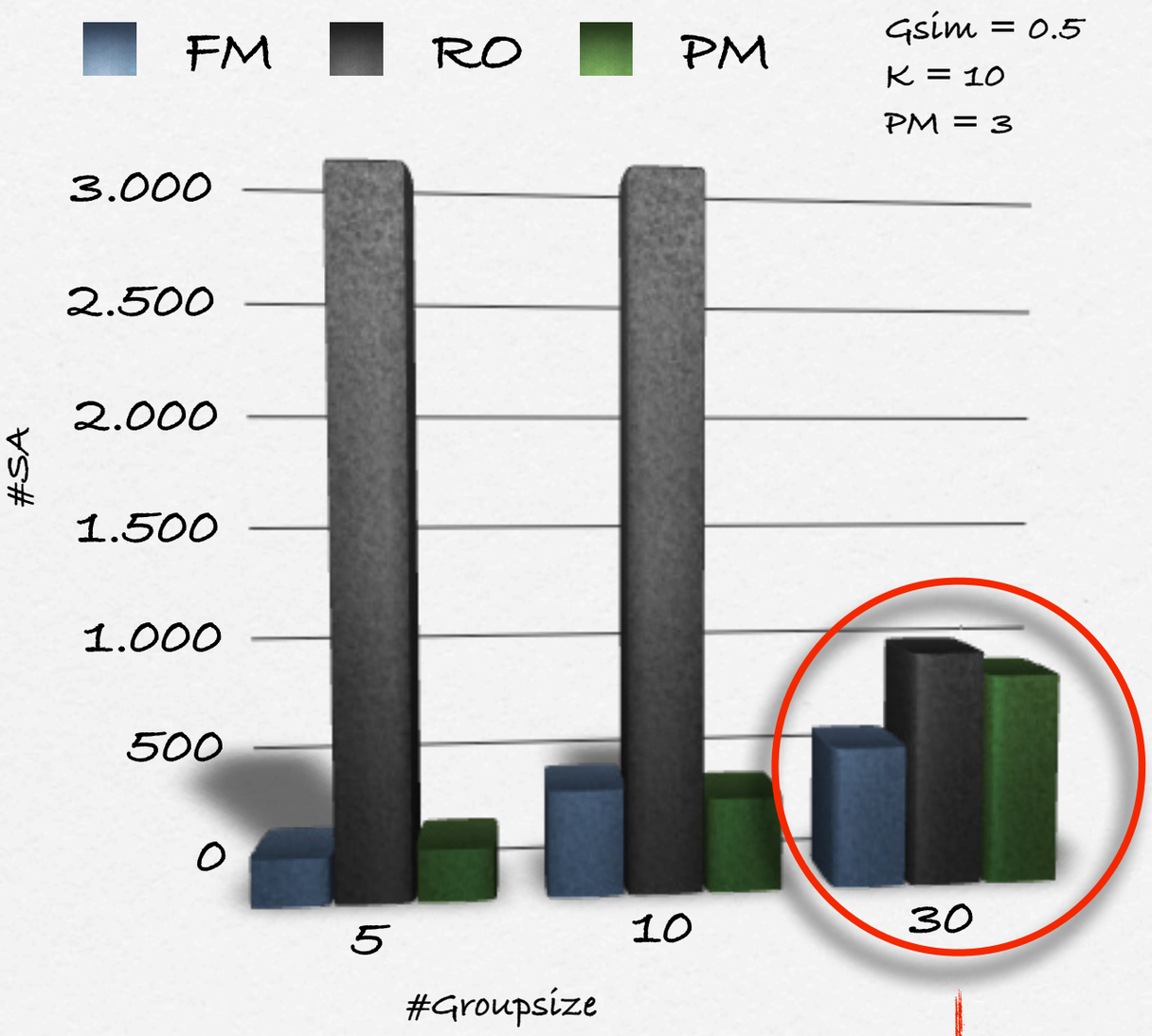
- o > similarity
- o < efficacia → o RO UB efficaci
- o Top item simili → o FM scansioni inutili

Varying no of items



- o > item
- o > #SA
- o Best: PM

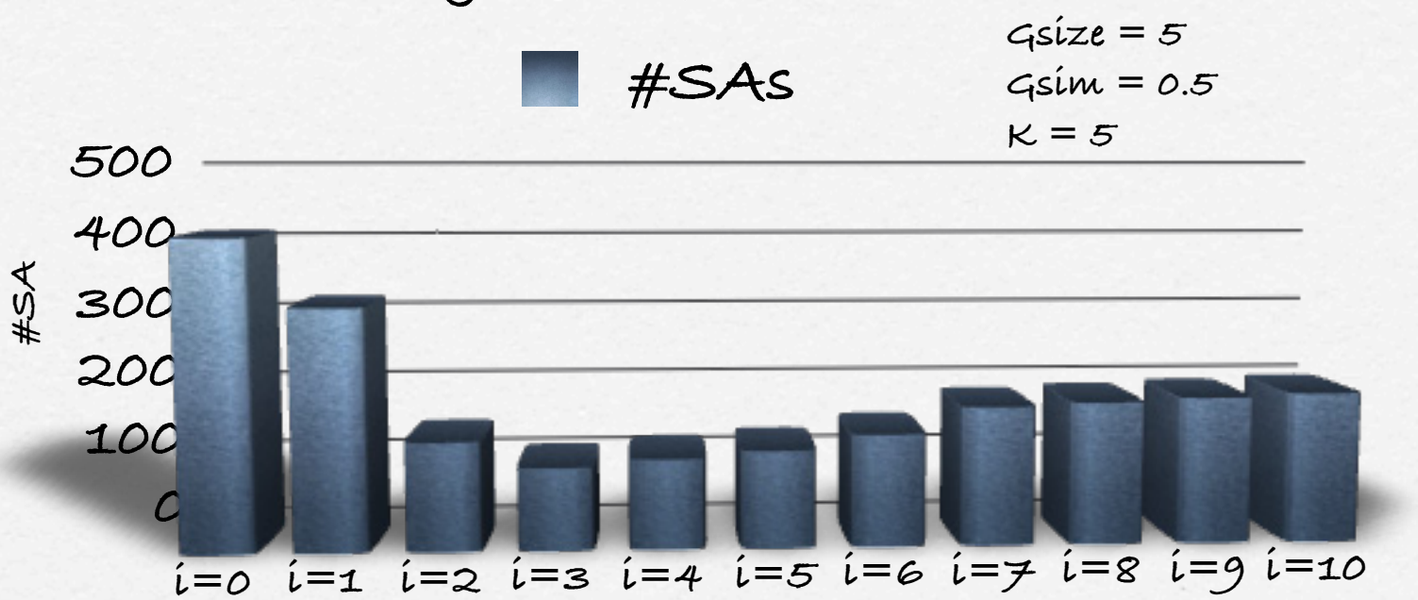
Varying Group Size



- 0 > #groupsize
- 0 > #SA
- 0 RO worst performance

IL forniscono buone "stopping condition"

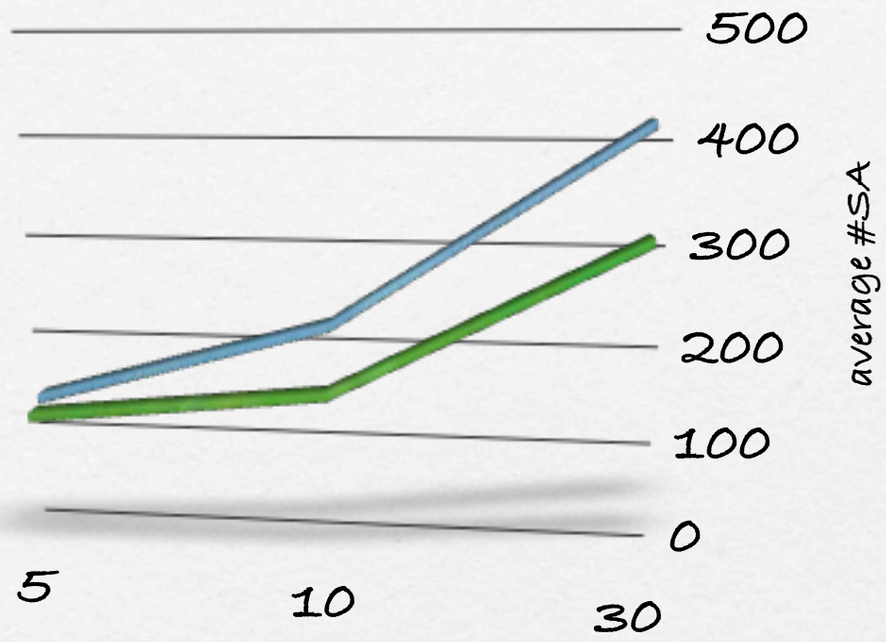
Varying # of materialized lists



- i=0 corrisponde a RO
- i=3 best performance

Trade-off tra #DL e #SA

Threshold Sharpening



- FM ottimizzato migliore

Riassumendo

- la presenza di Disagreement List migliora di molto le performance per gruppi di utenti "dissimilar" e di grandi dimensioni
- Spesso Partial Materialization (PM) è la soluzione migliore (gruppi "abbastanza" simili)
  - Per la stessa query, DL diverse contribuiscono diversamente nel calcolo del TOP-K
  - Solo alcune di esse devono essere considerate
- La Sharpening Optimization migliora sempre le performance complessive

Conclusioni

- Group Recommendation sono sempre più importanti grazie alle continue interazioni tra utenti sul WEB
- Disagreement List ha un forte impatto sia sulla qualità che sull'efficienza di Group Recommendation
- Threshold algorithm, TA può essere adattato per il calcolo di Group Recommendation. (FM, RO)
- Sono state implementate ottimizzazioni che migliorano l'efficienza degli algoritmi (PM, Sharpening Threshold)
- Lavori futuri: ottimizzare le DL affinché occupino meno spazio (contenendo le stesse info) e adattare gli algoritmi di GR

Conclusioni

Grazie dell'attenzione

Vota Gruppo 7