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

Lecture 2: Network-Aware Search in Social Tagging Sites

Sihem Amer-Yahia

Julia Stoyanovich

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

Summary of last lecture

- **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
- **Generalizations and extensions**

Quote of the day

A city is oneness of the unlike. ~Aristotle

Collaborative tagging sites

- **Social content sites are cyber-cities!**
- **Collaborative tagging sites are a kind of social content sites**
  - *Flickr*, *YouTube*, *Delicious*, photo tagging in *Facebook*
- **Users**
  - contribute content
    - annotate *items* (photos, videos, URLs, …) with *tags*
  - form social networks
    - friends/family, interest-based
  - consume content
    - browse own and other users' items
    - need help discovering **relevant** content
- **Goal**
  - Personalize search and information discovery

Outline

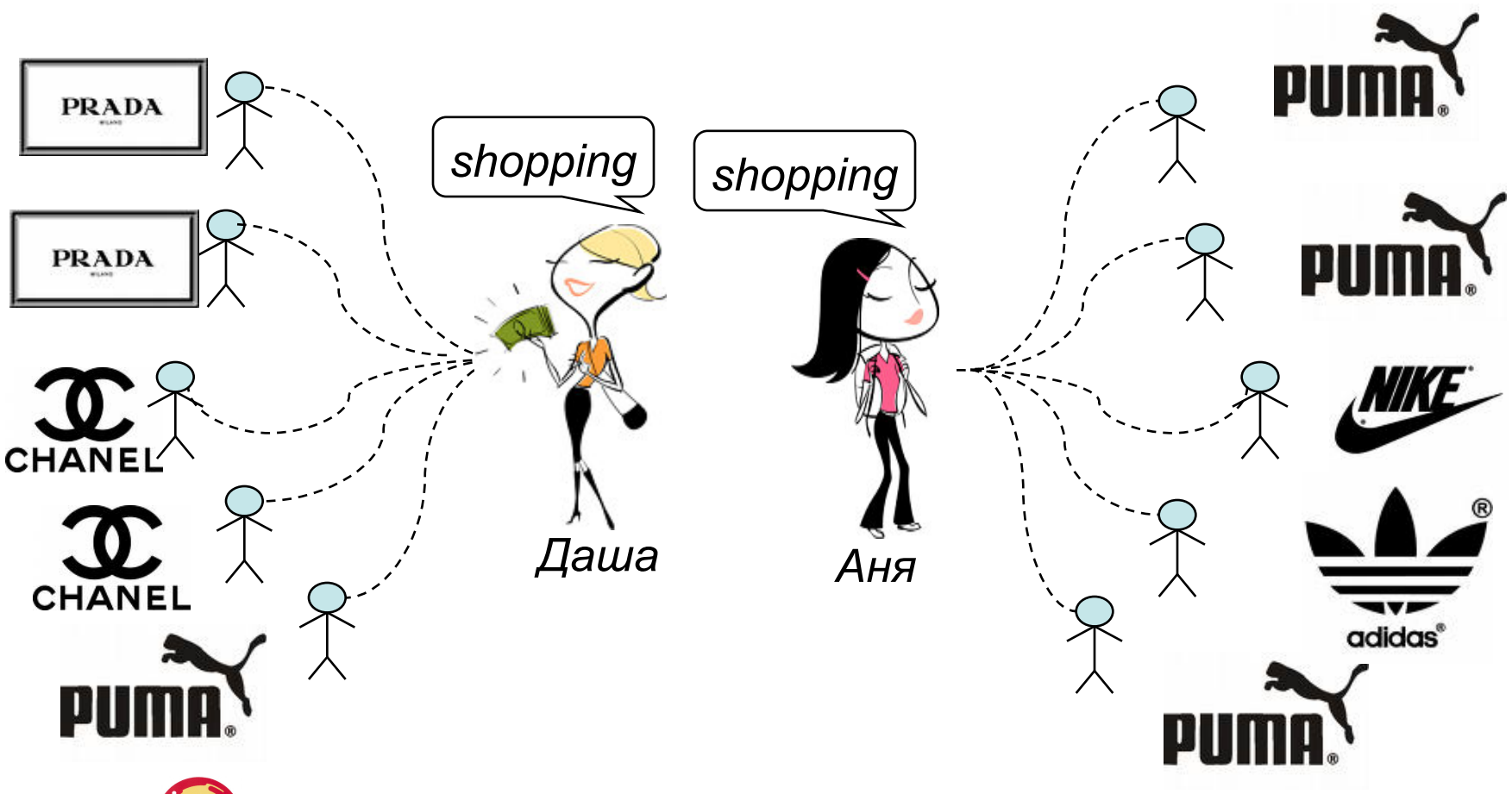
- ✓ Intro

- **Semantics**
  - Personalized ranking functions
  - Model and problem statement

- **Fundamental indexing structures and algorithms**
  - EXACT
  - Global Upper-Bound (GUB)
  - gNRA and gTA

- **Performance optimizations**
  - Cluster-Seekers
  - Cluster-Taggers

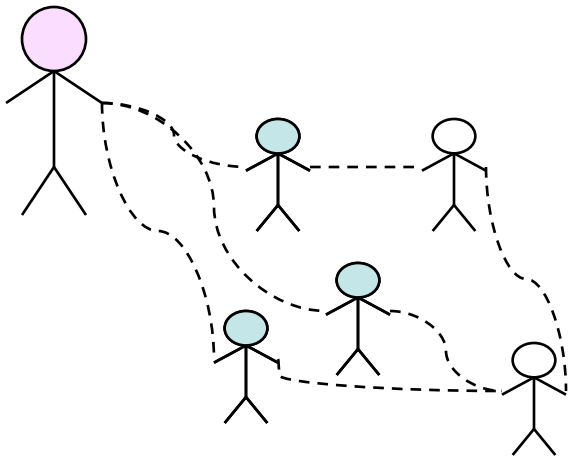
Why network-aware search?



Result relevance depends on who is asking the query!

Data model

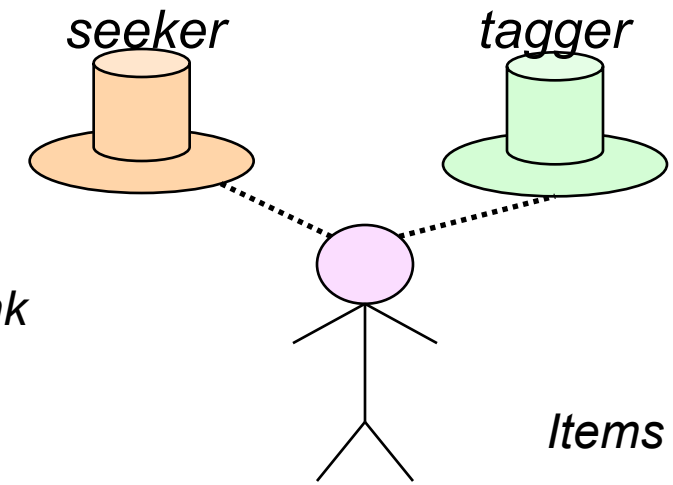
Link (user u, user v)



Network (u) =
{ v | Link (u,v) }

Tagged(user u,item i,tag t)

Roger, i1, music
Roger, i3, music
Roger, i5, sports
…
Hugo, i1, music
Hugo, i22, music
…
Minnie, i2, sports
…
Linda, i2, football
Linda, i28, news
…



Seekers = Π_{user} *Link*

Taggers = Π_{user} *Tagged*

Items (u) = Π_{item} ($\sigma_{user=u}$ *Tagged*)

Semantics of relevance

- **The system may derive any number of networks**
  - Are they useful?
  - Which of them are more useful than others?
- **Goal: capture user interests based on social behavior**
  - Tagging: an implicit social tie
  - Friendship: an explicit social tie
- **Validation: modeling tagging patterns in *Delicious*** ***[AAAI-SIP 2008]***
  - Is there over-all consensus on the tagging?
  - Is my tagging similar to my that of my friends?
  - Is my tagging similar to that of people who use the same tags as I do?
  - Is my tagging similar to that of people who tag the same items as I do?

Quantifying agreement between users

- **Let's forget about top-*k* for a second**
  - Consider *items(u)* and *items(v)* as sets
- **Directed**

$$agr(u,v) = \frac{|items(u) \cap items(v)|}{|items(u)|} \qquad agr(u,v) \neq agr(v,u)$$

- **Undirected (Jaccard similarity)**

$$agr(u,v) = \frac{|items(u) \cap items(v)|}{|items(u) \cup items(v)|} \qquad agr(u,v) = agr(v,u)$$

- **Many other options, we will focus on these two for simplicity**

Take 1: no need for personalization

Global top-10

Rank	URL	Votes
1	google.com	980
2	facebook.com	820
3	iTunes.com	729
4	twitter.com	720
5	jonasbrothers.com	680
6	cnn.com	678
7	amazon.com	620
8	yahoo.com	525
9	youtube.com	524
10	**techcrunch.com**	492

Items(Даша)

URL	Tag
jars.com	java
java.sun.com	java
techcrunch.com	news
devshed.com	tutorial

Items(Маша)

URL	Tag
bbc.co.uk	news
pbs.org	news
tomwaits.com	music
nick-cave.com	music
loureed.com	music

Quality: *coverage (`Global top-10`) = 3%*

Applicability: *scope (`Global top-10`) = 100%*

Take 2: account for tags only

Intuition: if a user tags with "music" she is interested in music

Items(Mawa)

URL	Tag
bbc.co.uk	news
pbs.org	news
tomwaits.com	music
nick-cave.com	music
loureed.com	music

Top-10 for "music"

Rank	URL	Votes
1	iTunes.com	542
2	eMusic.com	420
3	pandora.com	350
4	thebeatles.com	330
5	jonasbrothers.com	215
6	madonna.com	175
7	rhapsody.com	148
8	**tomwaits.com**	133
9	lastfm.com	120
10	beyonce.com	107

Top-10 for "news"

Rank	URL	Votes
1	cnn.com	610
2	**bbc.co.uk**	503
3	npr.org	427
4	nytimes.com	414
5	slashdot.org	392
6	reuters.com	330
7	news.cnet.com	290
8	msnbc.msn.com	250
9	news.yahoo.com	180
10	digg.com	149

1 tag	*coverage = 10%*	*scope = 32%*
2 tags	*coverage = 14%*	*scope = 14%*
3 tags	*coverage = 18%*	*scope = 6%*

Take 2: what's the problem?

Take 3: account for items only

Intuition: interests of users who tag similar items are similar

Items(Аня)

URL	Tag
bbc.co.uk	news
pbs.org	news
nytimes.com	news
nirvana.com	music
metallica.com	music
acdc.com	music
jars.com	work
techcrunch.com	work

Items(Даша)

URL	Tag
jars.com	java
java.sun.com	java
techcrunch.com	news
devshed.com	tutorial

Items(Маша)

URL	Tag
bbc.co.uk	news
pbs.org	news
tomwaits.com	music
nick-cave.com	music
nirvana.com	music

Items(Ваня)

URL	Tag
jars.com	work
java.sun.com	work
techcrunch.com	work
devshed.com	work
web2expo.com	work
technorati.com	work
javablogs.com	work
trenitalia.it	play

$$agr(u,v) = \frac{|items(u) \cap items(v)|}{|items(u)|}$$

Link (u, v, agr)

Аня,	Маша,	3/8
Маша,	Аня,	3/5
Ваня,	Даша,	1/2
Даша,	Ваня,	1
Аня,	Ваня,	1/4
Ваня,	Аня,	1/4
Аня,	Даша,	1/4
Даша,	Аня,	1/2

coverage up to 85%
but scope very low, about 1%

Other options?

- **Take 4:account for tags *and* items**
  - Intuition: multiple interests per user, overlap in items *per tag*

    coverage up to 82%
    scope up to 7%

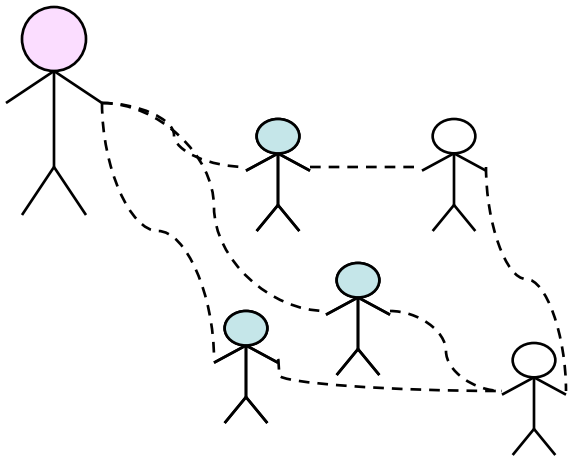
- **Take 5: account for friendship**
  - Intuition: interests of users a similar to those of their friends

    coverage = 43%
    scope = 31%

Social behavior (friendship and tagging) is reflective of a user's interests. That is, network-aware search makes sense.

Recall the data model

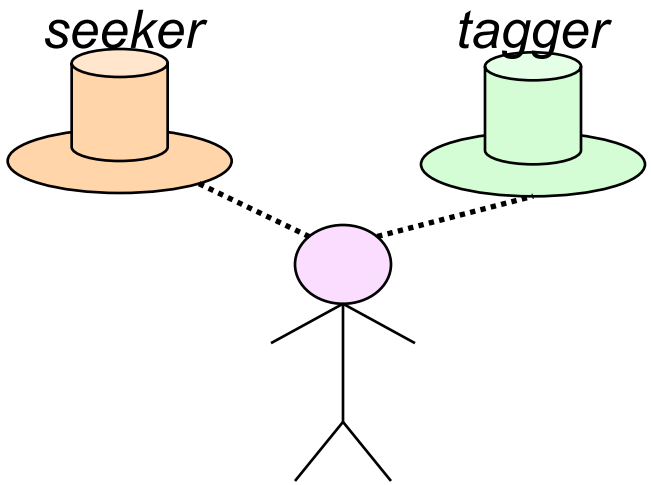
Link (user u, user v)



Network (u) =
{ v | Link (u,v) }

Tagged(user u,item i,tag t)

Roger, i1, music
Roger, i3, music
Roger, i5, sports
…
Hugo, i1, music
Hugo, i22, music
…
Minnie, i2, sports
…
Linda, i2, football
Linda, i28, news
…



Seekers = Π_u *Link*

Taggers = Π_u *Tagged*

Problem statement

- **A query is a set of *tags*** ***[VLDB 2008]***

  $Q = \{t_1, t_2, \ldots, t_n\}$

- **For a seeker *u,* a tag *t*, and a item *i***

  $score(i, u, t) = |\ Network(u) \cap \{v : Tagged(v, i, t)\}\ |$

  $score(i, u, Q) = score(i, u, t_1) + score(i, u, t_2) + .. + score(i, u, t_n)$

Given a query Q issued by a seeker u, we wish to efficiently determine the top k items, i.e., the k items with highest over-all score.

Outline

- ✓ Intro
- ✓ Semantics
  - ✓ Personalized ranking functions
  - ✓ Model and problem statement
- **Fundamental indexing structures and algorithms**
  - EXACT
  - Global Upper-Bound (GUB)
  - gNRA and gTA
- **Performance optimizations**
  - Cluster-Seekers
  - Cluster-Taggers

Recall standard top-*k* algorithms

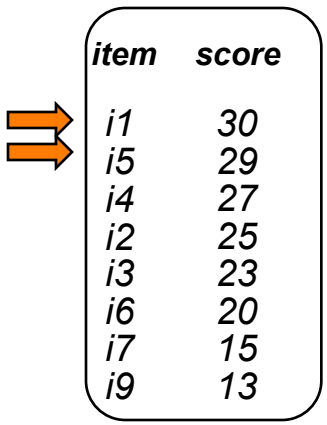
$Q = \{t_1, t_2, \ldots, t_n\}$; $score(i, Q) = score(i, t_1) + score(i, t_2) + \ldots + score(i, t_n)$

Indexing: per-tag inverted lists, each sorted on score

[PODS 2001]

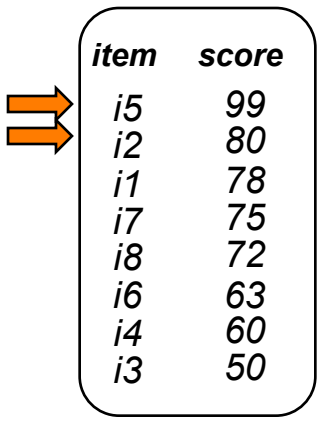
The *NRA* algorithm *(no random access)*

- access all lists sequentially, in parallel
- maintain a heap sorted on *partial* scores
- stop when score of k^{th} item $\geq$ sum of current list scores



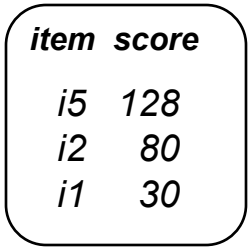
item	score
i1	30
i5	29
i4	27
i2	25
i3	23
i6	20
i7	15
i9	13

tag = shoes



item	score
i5	99
i2	80
i1	78
i7	75
i8	72
i6	63
i4	60
i3	50

tag = shopping



item	score
i5	128
i2	80
i1	30

top-K heap

$K = 1$

Stopping condition: $128 > 29 + 80$

Naïve solution: EXACT

- **Maintain single inverted list per (seeker, tag), items ordered by score**
  - **+ can use standard top-*k* algorithms**
  - -- high space overhead

Conservative example:

- *–100K users, 1M items, 1K tags*
- *–20 tags/item from 5% of the taggers*
- *–10 bytes per inverted list entry*
- ***–1 Terabyte of storage!***

Don't try this at home!

tag = shoes

item	score
i1	30
i8	29
i4	27
i2	25
i3	23
i6	20
i7	15
i9	13

seeker Даша

item	score
i5	99
i2	80
i8	78
i7	75
i1	72
i6	63
i4	60
i3	50

seeker Аня

tag = shopping

item	score
i1	73
i2	65
i3	62
i4	40
i5	39
i6	18
i7	16
i8	16

seeker Даша

item	score
i5	53
i9	36
i2	30
i6	15
i1	14
i8	10
i7	10
i3	5

seeker Аня

Exact scores vs. score upper-bounds

EXACT: 1 list per (seeker, tag)

item	exact score
i1	**73**
i2	**65**
i3	**62**
i4	**40**
i5	39
i6	**18**
i7	**16**
i8	**16**

seeker Даша

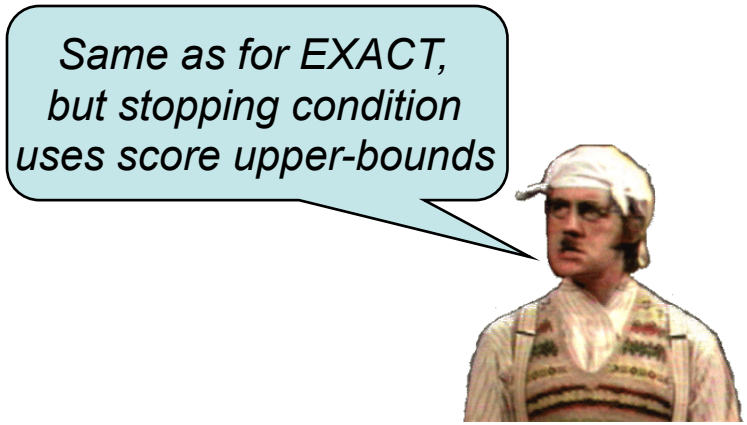
item	exact score
i5	**53**
i9	**36**
i2	30
i6	15
i1	14
i8	10
i7	10
i3	5

seeker Аня

Global Upper-Bound (GUB): 1 list per tag

item	taggers	upper-bound
i1	Miguel,…	73
i2	Kath, …	65
i3	Sam, …	62
i5	Miguel, …	53
i4	Peter, …	40
i9	Jane, …	36
i6	Mary, …	18
i7	Miguel, …	16
i8	Kath, …	16

both seekers



Top-*k* with score upper-bounds

score(i,u,t) = |Network(u) ∩ {v | Tagged(v,i,t)}|

ub(i,t) = max $_{u \in Seekers}$ ***score(i,u,t)***

gNRA - "almost no random access" generalization of NRA

- access all lists sequentially in parallel
- when an item is under the cursor, evaluate its *partial exact score*
- maintain a heap with *partial* exact scores
- stop when partial exact score of k^{th} item > **sum of current list upper-bounds**
- complete exact scores of top-*k* items on the heap using random accesses

gTA - generalization of TA

Example: gTA with GUB vs. with EXACT

tag = shoes

item	score
i1	30
i8	29
i4	27
i2	25
i3	23
i6	20
i7	15
i9	13

seeker Даша

item	score
i5	99
i2	80
i8	78
i7	75
i1	72
i6	63
i4	60
i3	50

seeker Аня

item	taggers	UB
i5	…	99
i2	…	80
i8	…	78
i7	…	75
i1	…	72
i6	…	63
i4	…	60
i3	…	50
i9	…	13

GUB

Q = "shoes shopping"

k = 3

Top-3 for Даша:

i1	103
i2	90
i3	85

Top-3 for Аня:

i5	152
i2	110
i8	88

When can we stop for each user with GUB?

When can we stop for each user with EXACT?

tag = shopping

item	score
i1	73
i2	65
i3	62
i4	40
i5	39
i6	18
i7	16
i8	16

seeker Даша

item	score
i5	53
i9	36
i2	30
i6	15
i1	14
i8	10
i7	10
i3	5

seeker Аня

item	taggers	UB
i1	…	73
i2	…	65
i3	…	62
i5	…	53
i4	…	40
i9	…	36
i6	…	18
i7	…	16
i8	…	16

GUB

Performance of GUB and EXACT

- **Evaluation on *Delicious*, 1 month worth of data**
  - 6 queries, 30 seekers per query, common interest network
- **Space overhead:** total # number of entries in all inverted lists
- **Query processing time:** # of cursor moves

	GUB	Exact
space (IL entries)	74K	63M
time	479-18K	13 - 189

space baseline

time baseline

Outline

- ✓ **Intro**
- ✓ **Semantics**
  - ✓ Personalized ranking functions
  - ✓ Model and problem statement
- ✓ **Fundamental indexing structures and algorithms**
  - ✓ EXACT
  - ✓ Global Upper-Bound (GUB)
  - ✓ gNRA and gTA
- • **Performance optimizations**
  - – Cluster-Seekers
  - – Cluster-Taggers

Clustering seekers

Global Upper-Bound

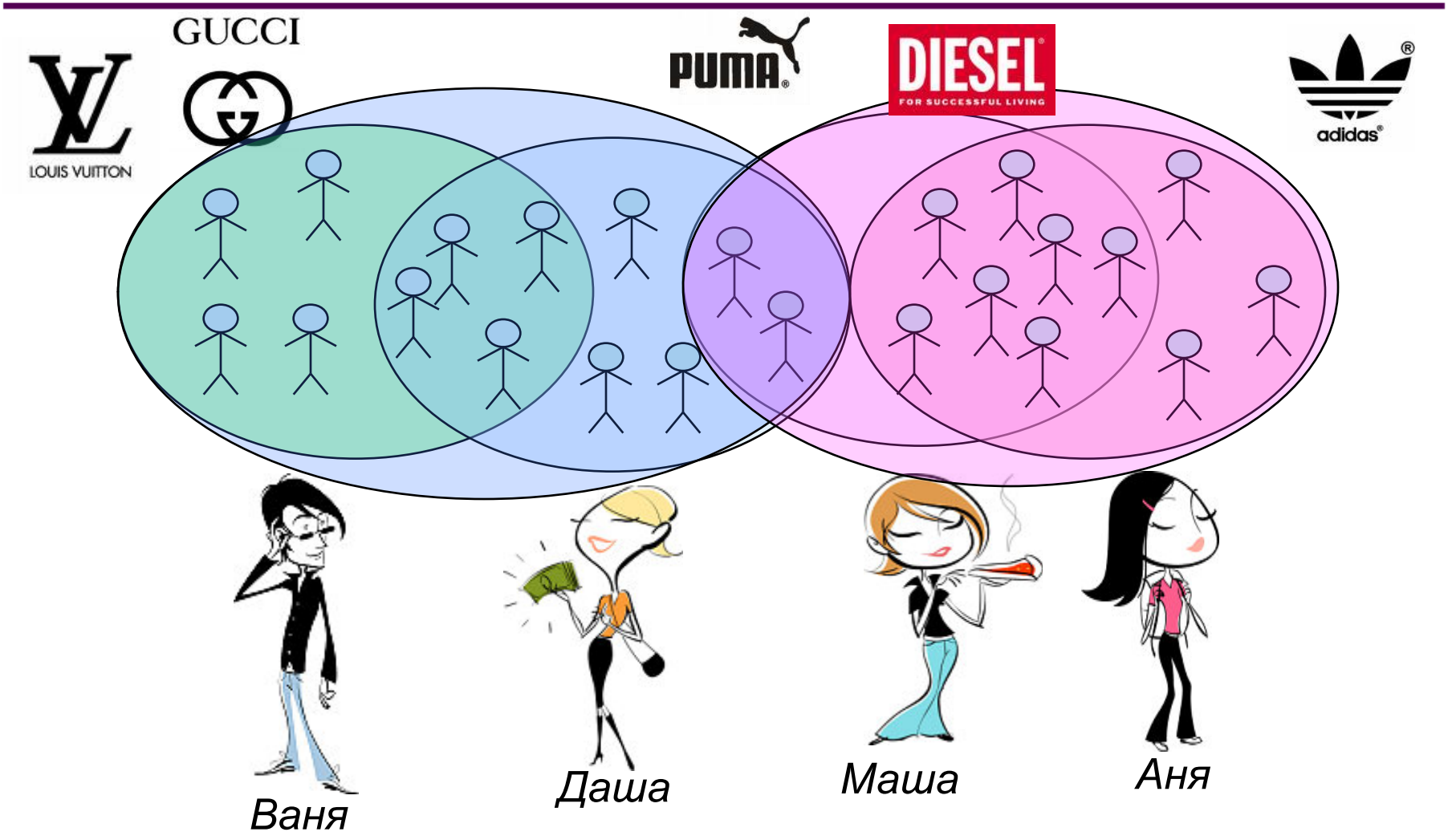
$$ub(i,t) = max_{u \in Seekers} score(i,u,t)$$

- **Problem: upper-bound order differs from exact score order for most users**
  - i.e. items that are most popular globally may not be most popular among particular networks for users (as we saw in Part 2 of the class)

Idea: cluster seekers based on *network overlap*

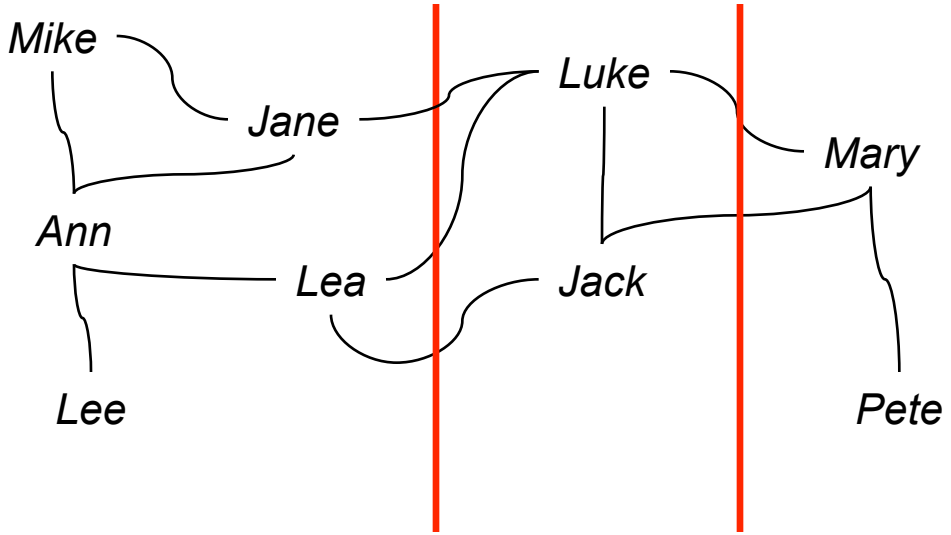
- score of an item for a seeker depends on the network
- if two seekers have overlapping networks -- they will have similar scores for many of the items

Seekers: network overlap

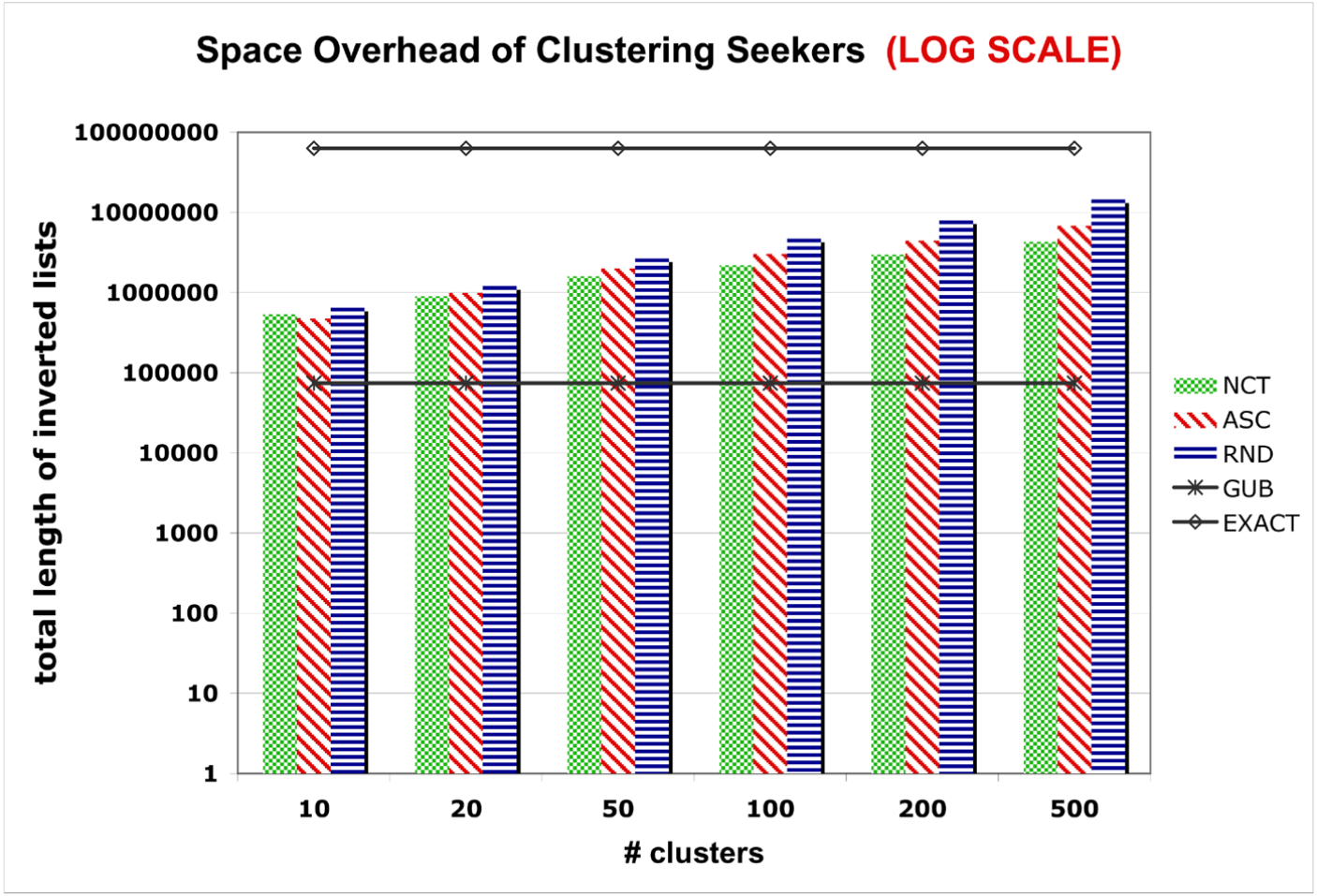


Clustering methods

- Clustered seekers independently for each tag
- Fix the number of clusters
- Use *Graclus* software package (University of Texas)
- **Random (RND)**: assign a seeker to a random cluster
- **Ratio Association (ASC)**: maximize edge density inside clusters
- **Normalized Cut (NCT)**: minimize edge-density across clusters



Cluster-Seekers: space

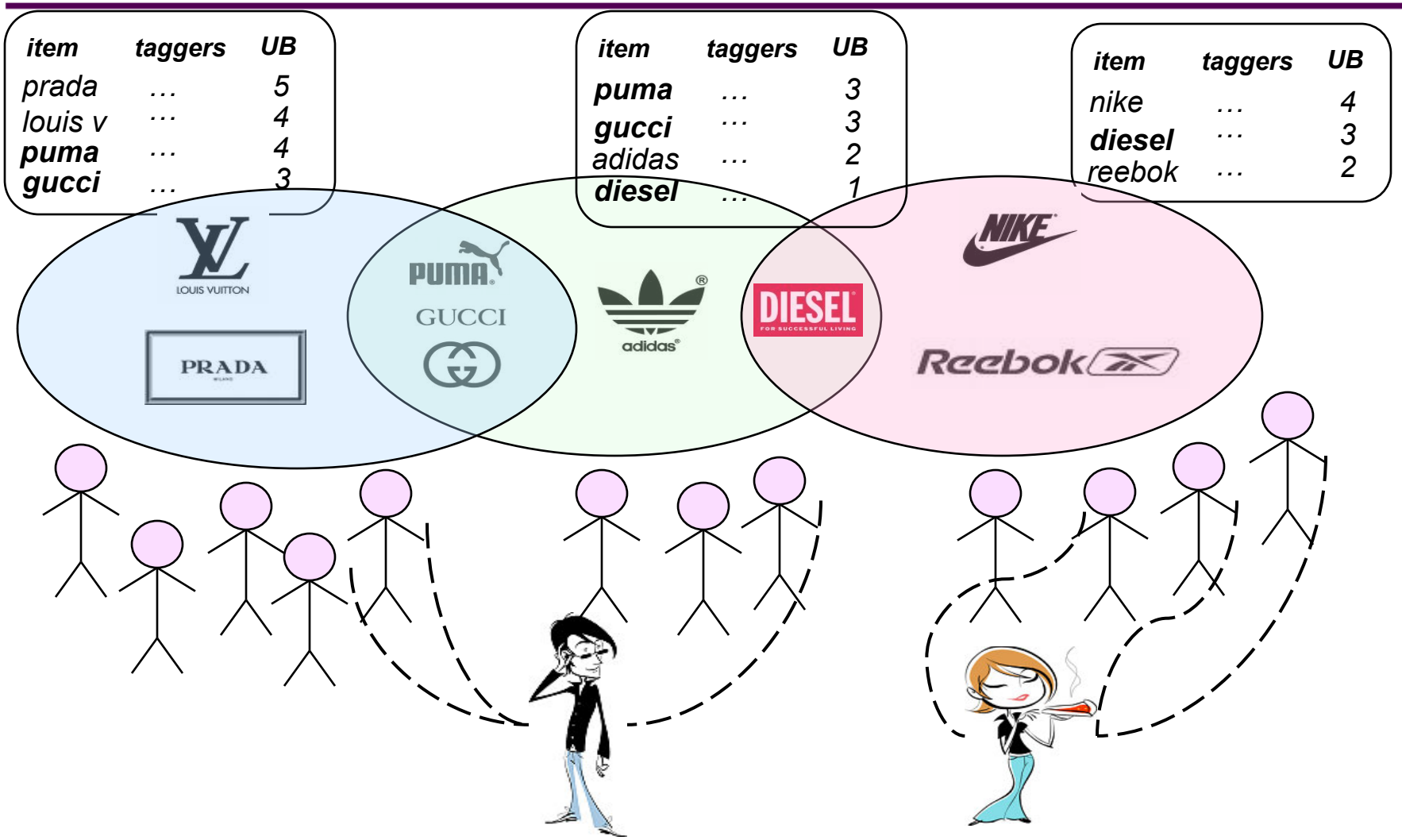


Cluster-Seekers*:* time

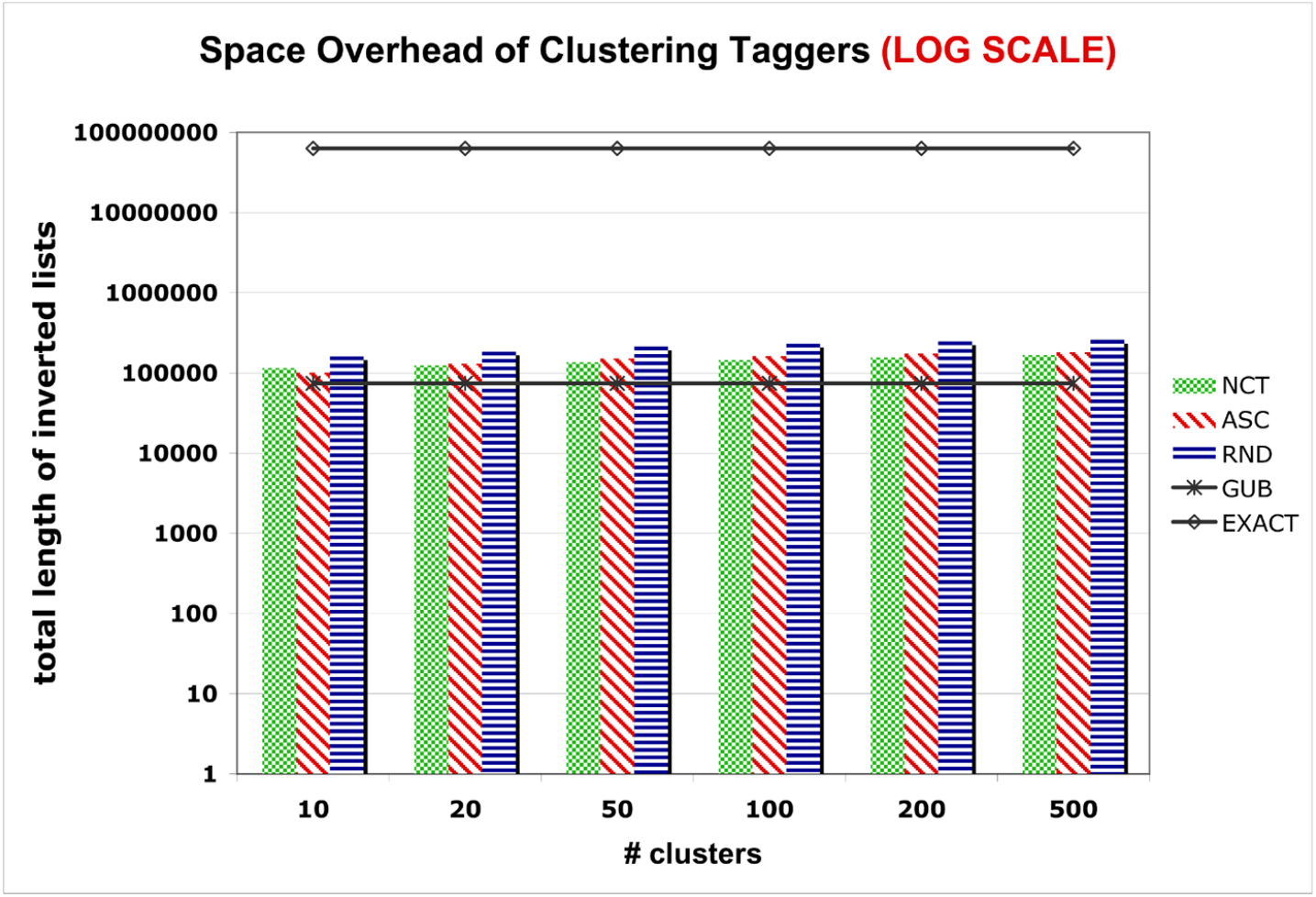
- ***Cluster-Seekers* improves execution time over *GUB* by at least an order of magnitude, for all queries and all users**
  - Inverted lists are shorter
  - *Score upper-bound* order similar to *exact score* order for many users

- **Average improvement between 38-87%**
  - Depends on the clustering method and on the number of clusters
  - Interestingly, normalized cut (NCT) has better space utilization, but ratio association (ASC) improves run-time performance more
  - Improvement even for a random clustering, why?

Clustering taggers: item overlap



Cluster-Taggers: space



Cluster-Taggers: time

- **We found that *Cluster-Taggers worked* best for seekers whose network falls into at most 3 * #tags clusters**
  - For others, query execution time degraded due to the number of inverted lists that had to be processed
- **For these seekers**
  - *Cluster-Taggers* outperformed *Cluster-Seekers* in all cases
  - *Cluster-Taggers* outperforms *Global Upper-Bound* by 94-97%, in all cases.

Discussion

- **Interesting follow-up work**
  - How to incorporate degree of friendship / network distance?
  - What about negative weights, can we accommodate these?
  - Do the performance results hold for different networks, different semantics of affinity? What would that depend on?

An alternative formulation

- **Alternative semantics – *holistic* personalized ranking** *[SIGIR 2008]*
  - Incorporate affinities between the seeker and taggers, e.g., friends, friends-of-friends, taggers who tag similarity
  - Incorporate personalized importance of tags for the seeker
  - Dynamically expand the query to similar tags
  - Combine score components using a *tf-idf* – style score (common in IR)
- **The ContextMerge algorithm**
  - Items(tag) – sorted on score-upper bounds for all users (like our GUB)
  - UserDocs(user), Friends(user), SimTags(tag)
  - Maintain upper / lower bounds for items; top-*k* and candidate heaps
- **Over-all**
  - The same motivation, but different ranking semantics, leading to a different technical approach
  - Processing could benefit from *Cluster-Seekers*

Summary and outlook

- **Semantics of personalized search in social tagging sites**
  - Exploring tagging and friendship to derive user affinity
- **Fundamentals of network-aware search**
  - Indexing structures: EXACT and global upper-bound
  - Top-*k* algorithms: gNRA and gTA
  - Time / space trade-off
- **Performance optimizations**
  - Cluster-Seekers: grouping seekers based on network similarity
  - Cluster-Taggers: grouping seekers based on item similarity
- **Next lecture**
  - Using top-*k* to generate recommendations for *groups of users*

References and further reading

1. *Optimal aggregation algorithms for middleware.*
   Ronald Fagin, Amnon Lotem and Moni Naor. PODS 2001.

2. *Leveraging tagging to model user interests in Delicious.*
   Julia Stoyanovich, Sihem Amer-Yahia, Cameron Marlow, Cong Yu.
   AAAI-SIP 2008.

3. *Efficient network-aware search in collaborative tagging sites.*
   Sihem Amer-Yahia, Michael Benedikt, Laks Lakshmanan, Julia Stoyanovich.
   VLDB 2008.

4. *Efficient top-k querying over social-tagging networks.* Ralf Schenkel and Tom Crecelius and Mouna Kacimi and Sebastian Michel and Thomas Neumann and Josiane Xavier Parreira and Gerhard Weikum. SIGIR 2008.

Questions?