Efficient Network-Aware Search in Collaborative Tagging Sites

Dogan Karaoglan

10 February 2009

UNIVERSITÄT DES SAARLANDES

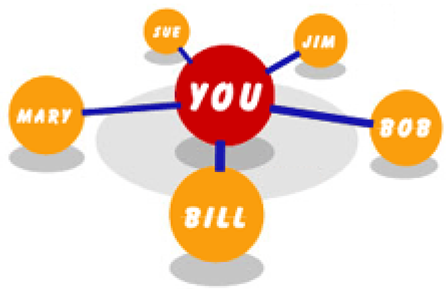
Talk Outline

- Introduction
- Collaborative Tagging Sites
- Data Model
- Problem Statement
- Top-k Processing
- Algorithmic Overview NRA
- Exact Scores & Score Upper - Bounds
- Experimental Evaluation
- Clustering Seekers
- Clustering Taggers
- Summary

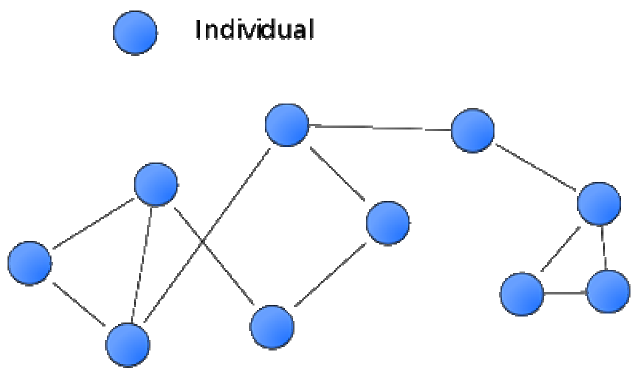
Introduction

- **Definition : Socail Network**

  A socail Network is a social structure made of nodes that are tied by one or more specific types of interdependency, such as values,



  - Visions
  - Ideas
  - financial exchange
  - friendship
  - kinship
  - ….



- In web people publish + tag information
- review + rate information
- publish their interests ….

Introduction

- What do we want to do ?

- Users see items and certain articles

- We want to search for efficient Objects

- Discovering relevant content

Collaborative Tagging Sites

- **Definition: Collaborative Tagging**

  Collaborative Tagging is free indexing of digital assets, in which the user on the basis of various social software applications, web pages using any number of keywords - called tags - labeled **.**

- **In Collaborative tagging sites**

  Publish/Subscribe both content and interest are dynamic

- **We model collaborative tagging sites**

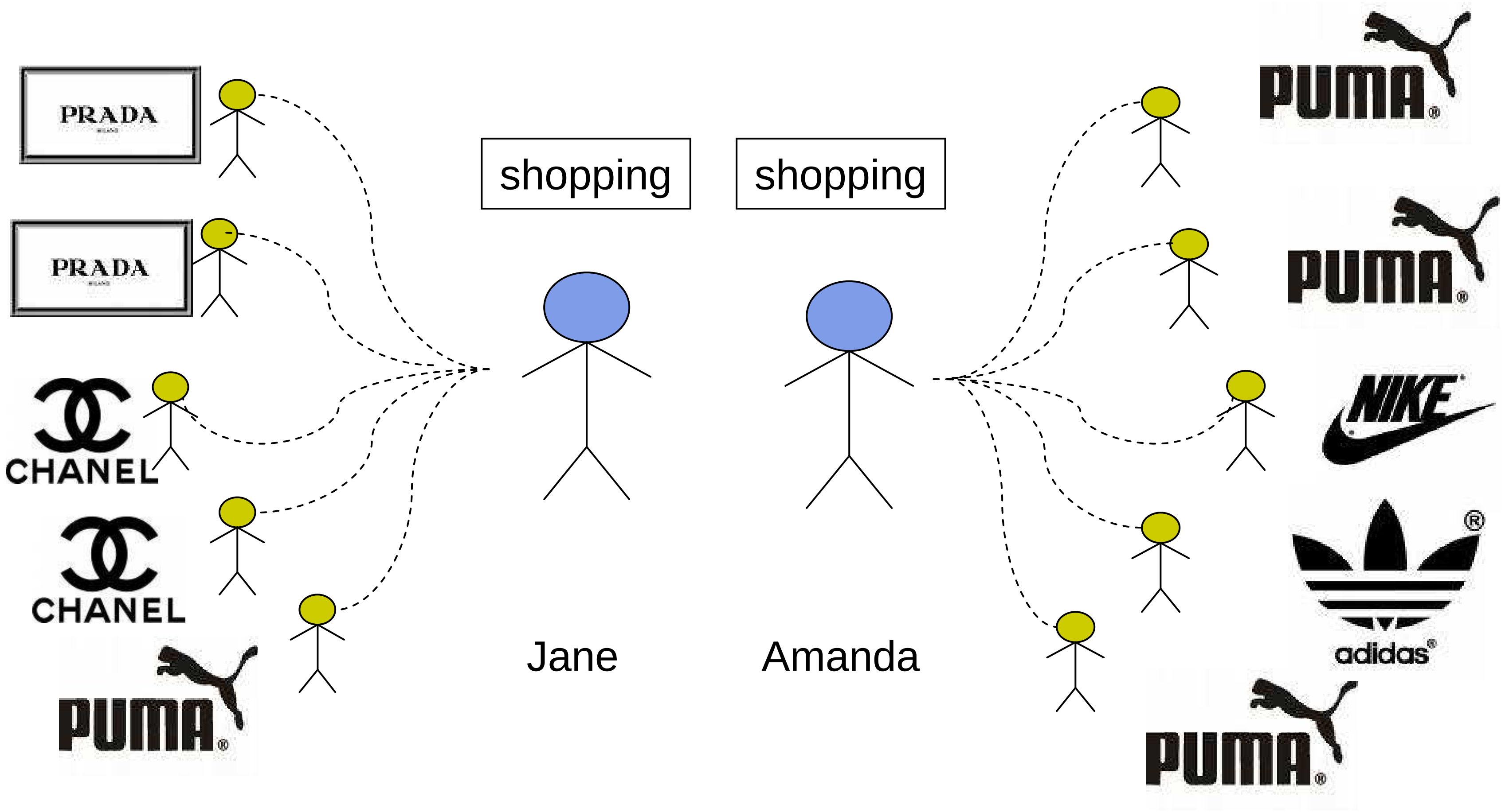
  Users in system = Taggers or Seekers

Collaborative Tagging Sites

- **Examples:**
  Flickr
  YouTube
  del.icio.us
  Facebook
- **Users**
  - contribute content
    - annotate *items* (photos, videos, URLs, Ideas…) with *tags*
  - form social networks
    - friends/family, interest-based
  - consume content
    - browse own and other users' items
    - need help discovering **relevant** content

Given a seeker , a network of taggers and a query
We wish to return the most relevant items.

Why Network-Aware Search ?



- Result relevance depend on who is asking the query!

Our Goals

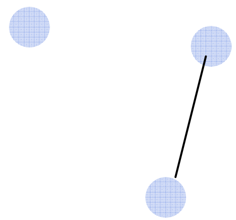
- Formalize the problem network-aware search

- Define and adapt top-k algorithms to Network-Aware Search, using score upper bounds

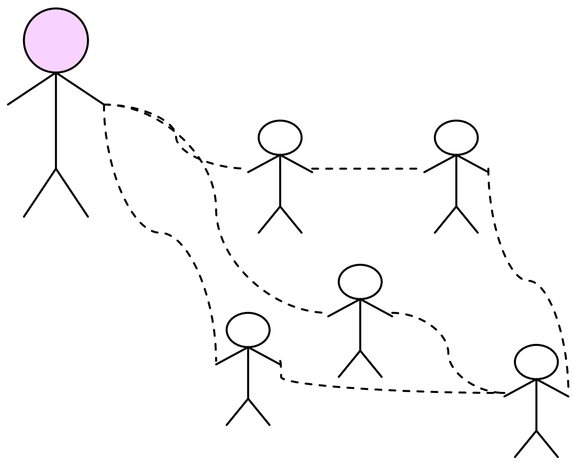
- Refine score upper-bounds based on the user’s network and tagging behavior

Data Model

- Graph Model
  Social Network as a directed Graph
  G=(V,E)
  Nodes are users
  Edges friendship, similar group ...
- Present the data relations



Link (user u, user v)



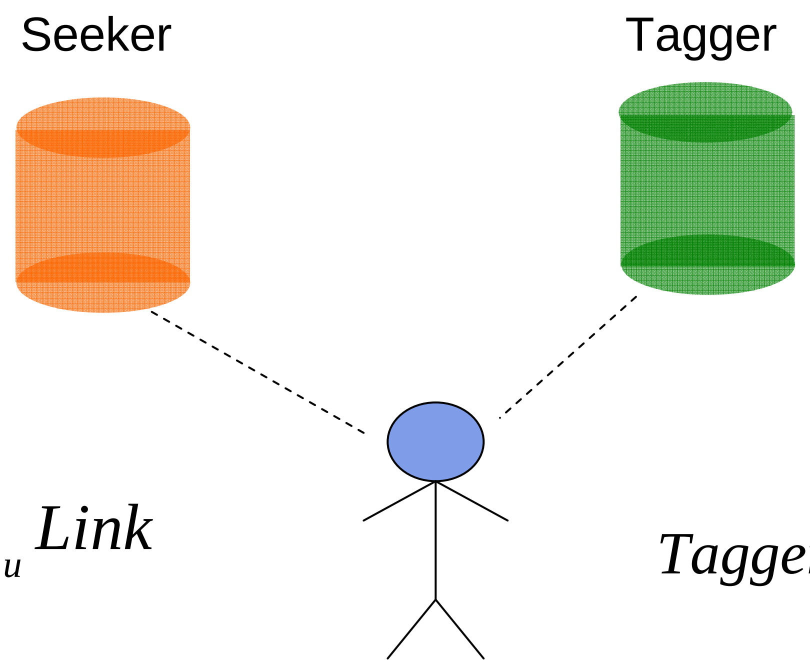
Tagged (user u ,item i ,tag t)

Roger, i1, music
Roger, i3, music
Roger, i5, sports
…
Hugo, i1, music
Hugo, i22, music
…
Minnie, i2, music
…
Linda, i25, news
Linda, i28, news
Miranda, i1, news

Data Model cont.

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

Items (v, t)={ i | Tagged (v, i, t) }



$$Seekers = \prod_u Link$$

$$Taggers = \prod_u Tagged$$

Problem Statement

- Given a query

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

- For a seeker u , tag t_j and a item i

We define a monotone function

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

$$score(i,u,Q) = g(score(i,u,t_1), scoreu(i,u,t_2), \ldots., score(u,i,t_n))\,|)$$

f and g are monotone, assume f= COUNT , g= SUM

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.*

Our Goals

- Formalize the problem network-aware search

- Define and adapt top-k algorithms to Network-Aware Search, using score upper bounds

- Refine score upper-bounds based on the user's network and tagging behavior

Top-k Processing

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

Indexing: inverted lists per tag, *IL1, IL2, … ILn,* sorted on scores

score (i) = g(score(i, IL1), score(i, IL2) , …, score(i, IL3))

Intuition: high-scoring items are close to the top of most lists

NRA *(no random access)*

- access all lists sequentially in parallel
- maintain a heap sorted on *partial* scores
- stop when partial score of *k* th item > best case score of unseen/incomplete items

Algorithmic Overview NRA

- Example: Top-1 for 2-term query (NRA)

L_1
A: 0.9
G: 0.3
H: 0.3
I: 0.25
J: 0.2
K: 0.2
D: 0.15

L_2
D: 1.0
E: 0.7
F: 0.7
B: 0.65
C: 0.6
A: 0.3
G: 0.2

top-1 item

min-k:

candidates

Algorithmic Overview NRA

- Example: Top-1 for 2-term query (NRA)

L_1	L_2
A: 0.9	D: 1.0
G: 0.3	E: 0.7
H: 0.3	F: 0.7
I: 0.25	B: 0.65
J: 0.2	C: 0.6
K: 0.2	A: 0.3
D: 0.15	G: 0.2

A: | 0.9 | ? |

score: [0.9;1.9]

top-1 item

min-k: 0.9

candidates

?: | ? | ? |

score: [0.0;1.9]

Algorithmic Overview NRA

- Example: Top-1 for 2-term query (NRA)

L_1	L_2
A: 0.9	D: 1.0
G: 0.3	E: 0.7
H: 0.3	F: 0.7
I: 0.25	B: 0.65
J: 0.2	C: 0.6
K: 0.2	A: 0.3
D: 0.15	G: 0.2

D: | ? | 1.0 |
score: [1.0;1.9]

top-1 item

A: | 0.9 | ? |
score: [0.9;1.9]

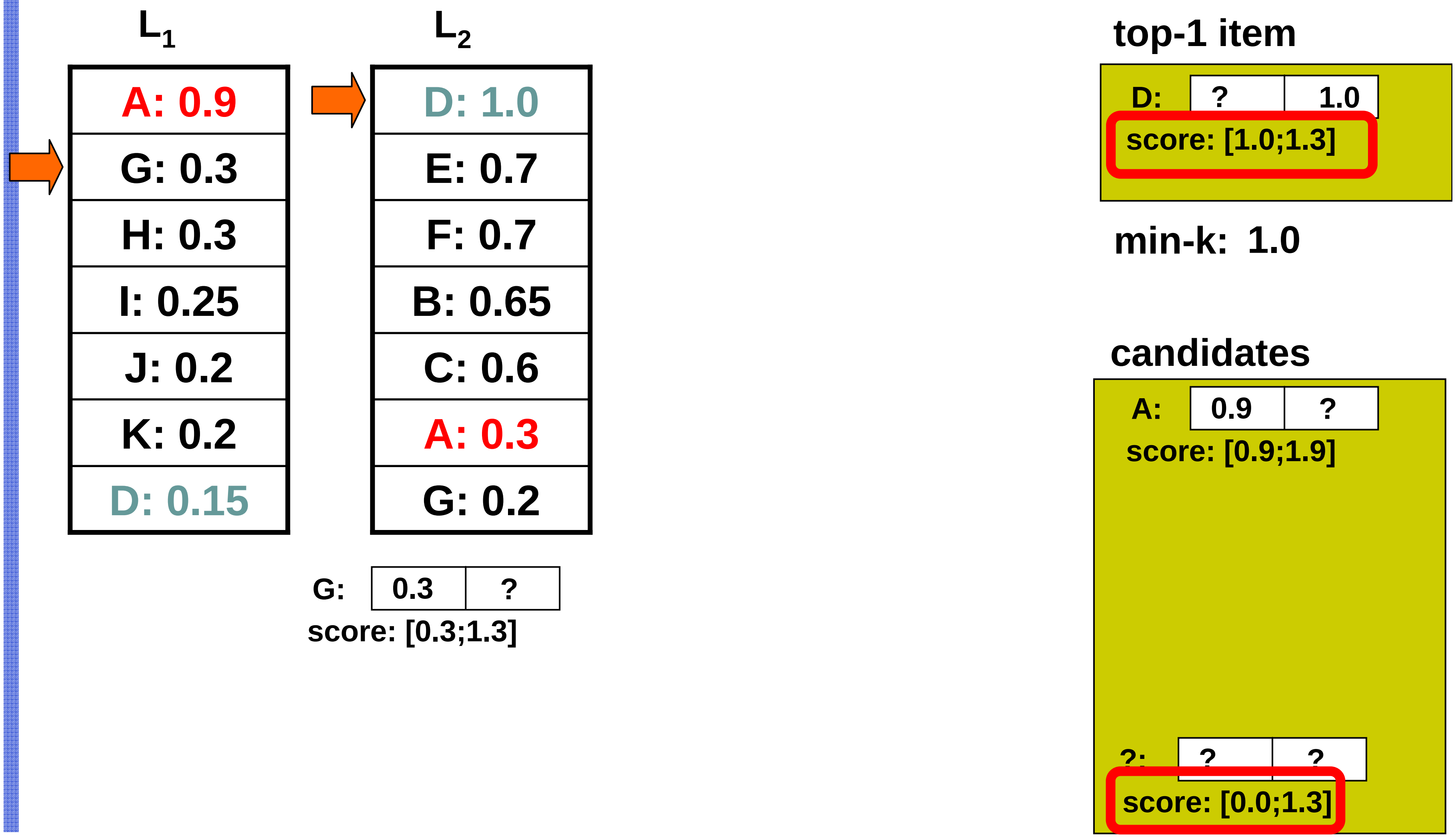
min-k: 1.0

candidates

?: | ? | ? |
score: [0.0;1.9]

Algorithmic Overview NRA

- Example: Top-1 for 2-term query (NRA)



Algorithmic Overview NRA

- Example: Top-1 for 2-term query (NRA)

L_1	L_2
A: 0.9	D: 1.0
G: 0.3	E: 0.7
H: 0.3	F: 0.7
I: 0.25	B: 0.65
J: 0.2	C: 0.6
K: 0.2	A: 0.3
D: 0.15	G: 0.2

No more new candidates considered

top-1 item

D:	?	1.0

score: [1.0;1.3]

min-k: 1.0

candidates

A:	0.9	?

score: [0.9;1.6]

~~G: 0.3 ? score: [0.3;1.0]~~

~~?: ? ? score: [0.0;1.0]~~

Algorithmic Overview NRA

- Example: Top-1 for 2-term query (NRA)

L_1	L_2
A: 0.9	D: 1.0
G: 0.3	E: 0.7
H: 0.3	F: 0.7
I: 0.25	B: 0.65
J: 0.2	C: 0.6
K: 0.2	A: 0.3
D: 0.15	G: 0.2

A:		
	0.9	0.4

score: [1.3;1.3]

Algorithm safely terminates

top-1 item

min-k: 1.3

candidates

~~D: ? 1.0~~

~~score: [1.0;1.2]~~

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!**

tag = *shoes*

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

seeker Jane

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

seeker Amanda

tag = *shopping*

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

seeker Jane

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

seeker Amanda

Exact Scores vs. Score Upper-Bounds

- Tag = shopping

EXACT: 1 list per (seeker, tag)

seeker Jane

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

seeker Amanda

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

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

both seekers

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

+low space overhead
-- item upper-bounds, and list order(!) may differ from EXACT for most users

Top-k with Score Upper-Bounds

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

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

gNRA *- "generalized no random access"*

- access all lists sequentially in parallel
- maintain a heap with *partial* exact scores
- stop when partial exact score of *k*th item > highest possible
- score from unseen/incomplete items (**computed using**
- **current list upper-bounds**

Experimental Evaluation

- Data
  - del.icio.us dataset, 1 month worth of data
  - Cleaned to remove some of the long tail
    - Removed items tagged by < 10 users
    - Removed tags used by < 4 users
  - 116K users, 176K items, 2.3M tagging actions, 903 tags

- Queries
  - 4 popular tags (in top-20)
  - 6 queries of length 2-4

- Users
  - *common-interest network:* link between a seeker and a tagger if they tagged at least 2 items in common
  - 30 seekers per query, with varying network characteristics (see paper for details)

Performance of GUB and Exact

- Space overhead
  - total # number of entries in all inverted lists
- Query processing time
  - # of cursor moves

Performance of GUB and Exact

- 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

Our Goals

- Formalize the problem network-aware search

- Define and adapt top-k algorithms to Network-Aware Search, using score upper bounds

- Refine score upper-bounds based on the user’s network and tagging behavior

Clustering Seekers

Global Upper-Bound

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

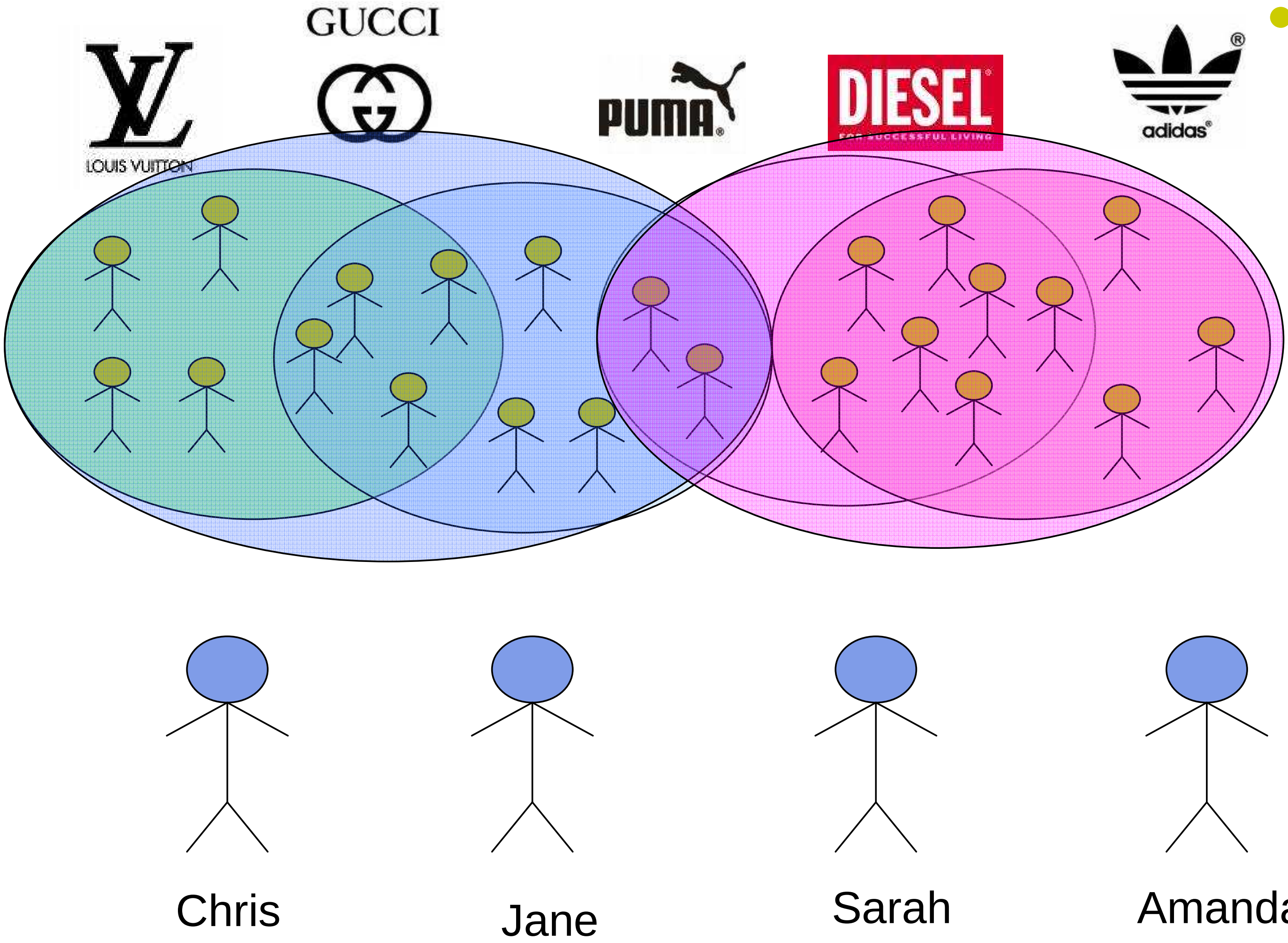
Problem: upper-bound order differs from exact score order for most users

- i.e. items that are most popular globally may not be most popular among particular networks for users

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 Seekers

Global Upper-Bound

item	*taggers*	*upper-bound*
puma	Miguel,…	73
gucci	Kath, …	65
adidas	Sam, …	62
diesel	Miguel, …	53
versace	Peter, …	40
nike	Jane, …	36
chanel	Mary, …	18
prada	Chris, …	16

Cluster-Seekers

item	*taggers*	*upper-bound*
gucci	**Kath**,…	65
versace	Peter, …	40
chanel	Mary, …	18
prada	Chris, …	16
puma	Peter, …	10

cluster 1: seekers Chris & Jane

item	*taggers*	*upper-bound*
puma	Miguel,…	73
adidas	Sam, …	62
diesel	Miguel, …	53
nike	Jane, …	36
gucci	**Kath**, …	5

cluster 2:seekers Amanda & Sarah

- assign each seeker to a cluster
- compute an inverted list per cluster

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

\+ tighter bounds, item order usually closer to EXACT order than in Global Upper-Bound

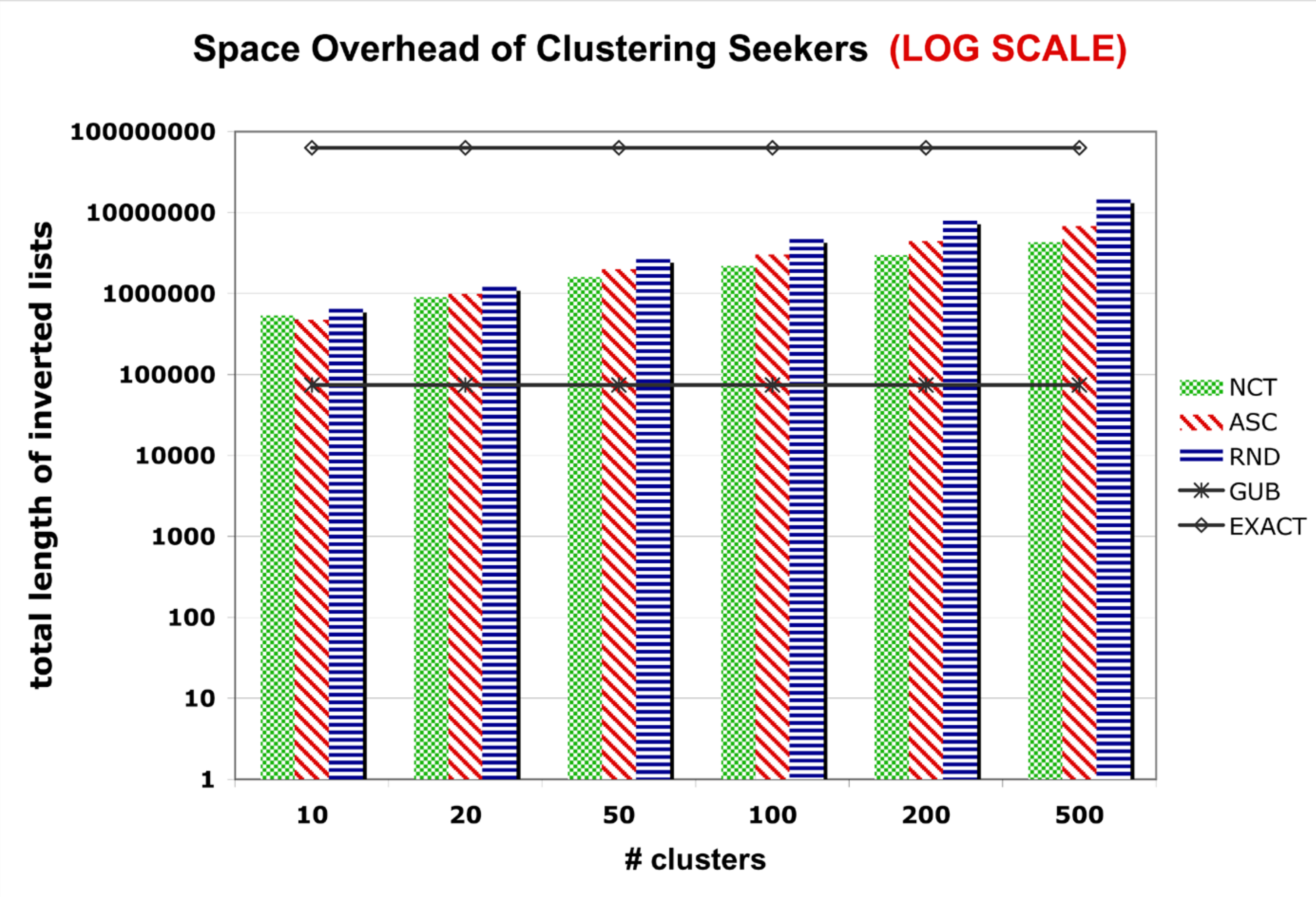
-- space overhead still high (trade-off)

Cluster-Seekers: Details

- Implementation
  - Java, Oracle 10g back-end for gNRA
  - *graclus* (U Texas) for clustering
    - Ratio Association (ASC) - maximize intra-cluster edge density
    - Normalized Cut (NCT) - minimize edge-density across clusters
  - Random cluster assignment (RND) as a clustering baseline

- Clustered seekers independently for each tag

Cluster-Seekers: Space



Cluster-Seekers : Time

- *Cluster-Seekers* improves query 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 over Global Upper-Bound
  - Normalized Cut: **38-72%**
  - Ratio Association **67-87%**

Cluster Taggers

- *Cluster-Seekers* problem: tagging actions of a single tagger may be replicated across multiple clusters
- Idea: cluster taggers based on overlap in tagging
  - assign each **tagger** to a cluster
  - compute cluster upper-bounds:

$$ub(i,t,C) = \max_{u \in Seekers, v \in V} |Network(u) \cap \{Tagged(v,i,t_j)\}|$$

\+ low space overhead

-- a seeker may map to multiple clusters, so more lists to process at query time

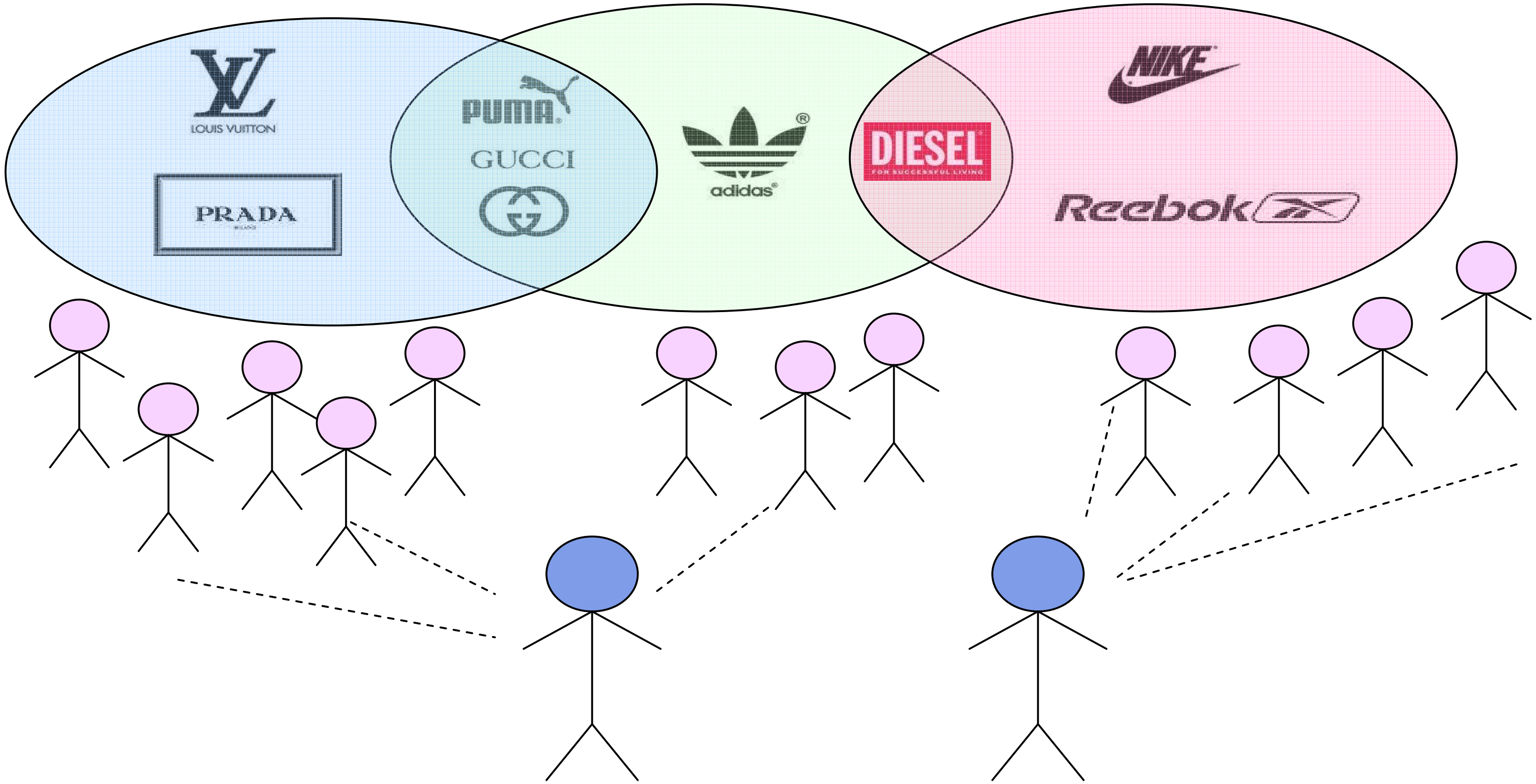
Taggers: Time Overlap

tag = *shopping*

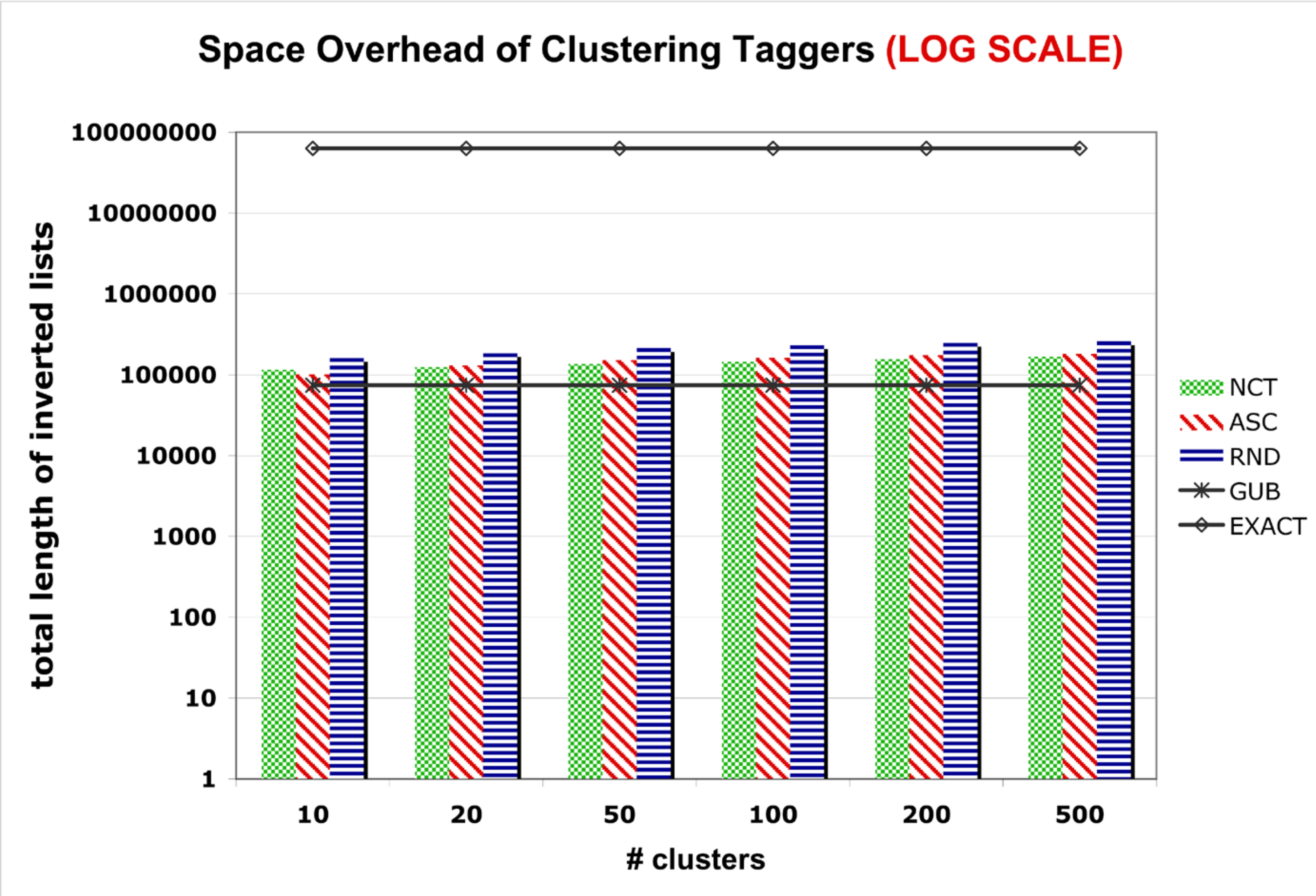
item	*taggers*	*UB*
prada	…	5
louis v	…	4
puma	…	4
gucci	…	3

item	*taggers*	*UB*
puma	…	3
gucci	…	3
adidas	…	2
diesel	…	1

item	*taggers*	*UB*
nike	…	4
diesel	…	3
reebok	…	2



Cluster-Taggers: Space



Cluster-Taggers: Time

- We found that *Cluster-Taggers worked* best for seekers whose network fell 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.

Clustering Quality

- Can we find perfect clusters?
- Answer : No, but we can try !

- Theorems: Finding a clustering that minimizes worst, average computation time of our top-*k* algorithms is NP-hard.
  - Proofs by reduction from *independent task scheduling* problem and *minimum sum of squares* problem

- Clustering quality can be tuned heuristically
  - Use a variant of Normalized Discounted Cumulative Gain (NDCG)
  - The metric compares the *ideal* (exact score) order in inverted lists with *actual* (score upper-bound) order

Summary

- Presented network-aware search in social tagging sites
- Extended top-*k* algorithms to work with score upper-bounds
- Proposed clustering of users that balances space consumption against query processing time
- Presented an evaluation on del.icio.us, a real social tagging dataset

- Tank you