

Social tagging systems

Ilaria Bordino

Department of Computer and Systems Science
Sapienza University of Rome
Rome, Italy
and

Department of Information and Communication Technologies
Universitat Pompeu Fabra
Barcelona, Catalunya, Spain

April 7th, 2010



Tagging systems

- Tagging systems allow users to annotate Internet resources (images, pages, videos) with keywords (tags) without relying on a controlled vocabulary.
- Tagging systems have the potential to improve search, spam detection, reputation systems, and personal organization.
- At the same time, they introduce new modalities of social interactions and opportunities for data mining. This potential is due to the social structure that underlies many of these systems.

Web-based tagging systems: Del.icio.us

The fastest bookmarks on the web.
Save your own or see what's fresh now!

Learn More

Search the biggest collection of bookmarks in the universe... Search

Hide Intro

Fresh Bookmarks | Hotlist | Explore Tags

The freshest bookmarks that are flying like hotcakes on Delicious and beyond.

See more recent bookmarks

New bookmarks saved in the last minute: 275

- 25 Dead in Blast at West Virginia Coal Mine - NYTimes.com** SAVE
via nytimes.com 10
news coalmine accident mine tragedy
- 'Oriental yeti' discovered in China - Telegraph** SAVE
via telegraph.co.uk 17
china yeti cryptozoology science iso3166-cn
- US court rules against FCC on 'net neutrality' - Yahoo! Finance** SAVE
via finance.yahoo.com 6
fcc internet 040510 obamaadmin ping.fm

Web-based tagging systems: Technorati

The screenshot displays the Technorati website interface. At the top, there is a search bar with the text "Search for posts..." and a magnifying glass icon. To the right of the search bar are links for "Join / Sign In / Help". Below the search bar is a navigation menu with categories: Technology, Business, Entertainment, Lifestyle, Sports, Politics, Videos, Blogging, and Twittorati. There is also a link to "Follow @trarticles On Twitter".

Below the navigation menu is a secondary menu with links: Blog Directory, Top 100, Tags, People, Technorati Blog, Write for Technorati, State of the Blogosphere, Android, and Guru. Underneath this is a section for "Ads by Google" with links to "Blog Directory", "Video Blog", "Blogger Photos", and "Blog Gratuito".

The main content area is titled "Today on Technorati" and features several article listings:

- Socially Networking Your Professional Reputation At Unvarnished.com**: A post from Unvarnished.com that looks to make recommendations and personal reviews a little more social and some say, a bit too social. (in Blogging)
- You've Got Your iPad Now Grab Some Apps**: A list of iPad media apps that will impress your friends and depress your enemies. (in Technology)
- Obama's Health Care Logorreakh**: President Obama tries unique tactic to quiet his health care reform critics: boring them to death. (in Politics)
- What's The 411 on Butler University?**: This year's NCAA Cinderella team is going all the way to the championship against...

On the right side of the page, there is a large advertisement for "Life In The Ocean Depends On You" with the slogan "Keep It Clean" and the website "www.KeepOceansClean.org". Below the ad is a section titled "Recently updated features" with items like "Chicks Dig the Longball" and "Apple's iPad".

Web-based tagging systems: FlickrR

flickr[®] from YOUSOO


Home The Tour Sign Up Explore


You aren't signed in Sign In Help


Search


Jump to: fresco

Explore / Tags / fresco / clusters

 [Italy, italia, rome, roma, pompeii, vascan, affresco, chiesa, europe, roman](#)
→ See more in this cluster...

 [church, painting, art, ceiling, architecture, mural, dome, turkey, wall, orthodox](#)
→ See more in this cluster...

 [fresh, water, green, verde, aqua, summer, acqua, estate, fruit, food](#)
→ See more in this cluster...

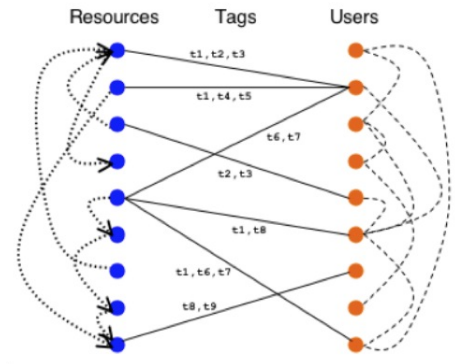
 [florence, firenze, tuscany, duomo, toscana, cathedral](#)
→ See more in this cluster...

What are social tagging systems useful for?

- Personal bookmarking. Tags allow users to collect, store and retrieve resources using the tags applied.
- Social-interaction features allow to connect individual bookmarking activities to a rich network of tags, resources, and users.
- Social tagging systems allow users to share their tags for particular resources. Each tag serves a link to additional resources tagged the same way by other users.

- No predefined taxonomy structure
- Social tagging systems rely on:
 - emergent social structures and behaviors;
 - related conceptual and linguistic structures of the user community;
- The popular tags form a **folksonomy**, a folk taxonomy of important and emerging concepts within the user group.
- Benefits: enhance the metadata for all users, and distribute the workload for creating such data.

Users, tags and resources



- search and information retrieval;
- information discovery, organization and communication;
- spam filtering, reducing effects of link spam, and improving on spam metrics;
- identifying trends and emerging topics globally and within communities;
- locating experts and opinion leaders in specific domains.

Two taxonomies for social tagging systems

- 1 System design and attributes
- 2 User incentives

System design and attributes

Tagging rights: the restrictions imposed on group tagging.

- 1 Self-tagging: the user can only tag the resources they created (e.g., Technorati);
- 2 Free-for-all tagging: any user can tag any resource, like Yahoo! Podcasts.

Systems allow various levels of compromise:

- systems can choose the resources that users are allowed to tag;
- they can specify different level of permissions to tag (flickr)
- they can specify who can remove a tag;

Free-for-all tagging systems are broad, both in the magnitude of the group of tags assigned to a resource, and in the nature of the tags generated.

System design and attributes

Tagging support:

- 1 blind tagging: a tagging user cannot view the tags assigned to the same resource by other users while tagging (e.g. del.icio.us);
- 2 viewable tagging: the user can see the tags already assigned to a resource (e.g., Yahoo! podcasts);
- 3 suggestive tagging: the system suggests possible tags to the users, based on
 - tags used by the same user
 - tags assigned to the same resource by other users
 - automatically gathered contextual metadata
 - machine-suggested tag synonyms.

Suggestive tagging may lead to a quicker convergence to a folksonomy.

System design and attributes

Aggregation of tags around resources

- 1 Bag model: the system allows a multiplicity of tags, and repetitions of tags from different users.
- 2 Set model: tag repetitions are not allowed. A group is asked to collectively tag a given resource.
- 3 In the first case the systems may maintain a richer information, and compute aggregate statistics to provide users with the breadth of opinions of the taggers.

- Type of object
- Source of materials: resources can be provided by
 - the users
 - by the system
 - a system can be open to the tagging of any web resource.

System and design attributes

- Resource connectivity: linked, grouped, or none. Web pages are connected by direct links; Flickr photos can be assigned to groups; events may have connections based on time, city and venue of the event.
- Social connectivity: some systems allow users to be connected with each other. User connectivity can be linked, grouped, or none.
- Several dimensions to consider: typed/directed links.
- Implication: adoption of localized folksonomy based on the underlying social structure of the system.

User incentive and behavior

- Users are motivated both by personal needs and by sociable interests.
- A large part of the motivations and influences of users is determined by the system designed and the way they are exposed to tagging practices.
- Users may also be persuaded by the norms of their friends and how they think that a system fits into their use.
- Tagging can be a public and sociable activity, but not all the tags do emerge with an intended audience.
- Why do people contribute and what are the results on the output and performance of the system?
 - At a very high level, the motivation to tag can be categorized into **organizational** or **social**.

Motivations for tagging behavior

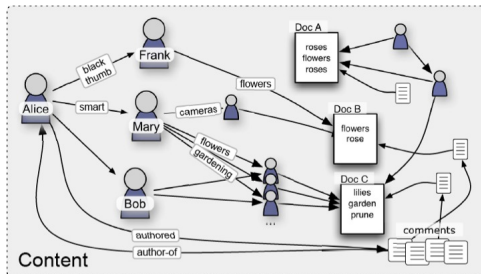
- 1 Future retrieval
- 2 Contribution and share
- 3 Attract attention
- 4 Play and competition
- 5 Personal presentation
- 6 Opinion expression

Challenges in Searching Online Communities

- Online content sites are characterized by an effective integration of the user's social network into the experience of exploring and tagging content.
- Online communities have a rich body of data comprised of user-distributed content, user relationships, and user ratings.
- Ranking of search results needs to take into account **social activity**.
- The results must be personalized based on the user's context.
- **Recency** is an important factor that requires dynamic incorporation of new content in a manner similar to news search.

Relevance factors for social content search

- Target resources: **Content**, hot keywords and people.

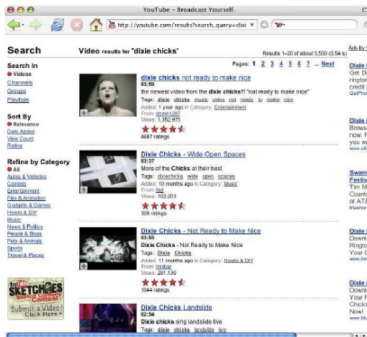


Relevance factors for social content search

- Text features
- Timeliness and freshness
- Incoming links and tags
- Popularity
- Social distance
- Relevance for people

Examples of social search

Keyword search



The screenshot shows a web browser window displaying YouTube search results for the keyword "dixie chicks". The browser's address bar shows the URL "http://youtube.com/results?search_query=dixie". The search results are displayed in a list format, with each entry including a video thumbnail, the video title, duration, description, upload date, category, and a star rating. The first result is "dixie chicks not ready to make nice" (4:59), the second is "Dixie Chicks - Wide Open Spaces" (3:27), the third is "Dixie Chicks - Not Ready to Make Nice" (4:16), and the fourth is "Dixie Chicks Landslide" (3:54). On the left side of the page, there are navigation options such as "Search in" (Videos, Channels, Groups, Playlists) and "Sort by" (Relevance, Date Added, View Count, Rating). On the right side, there are additional search filters and a search history section.

YouTube - Broadcast Yourself

http://youtube.com/results?search_query=dixie

Search Video results for "dixie chicks" Results 1-20 of about 5,550 (of 54)

Page: 1 2 3 4 5 6 7 - Next

Search in

- Videos
- Channels
- Groups
- Playlists

Sort by

- Relevance
- Date Added
- View Count
- Rating

Refine by Category

- All
- Music & Videos
- Comics
- Entertainment
- Fun & Imagination
- Games & Children
- Health & Diet
- Home
- News & Politics
- People & Blogs
- Pets & Animals
- Sports
- Travel & Places

SKETCHES

Submit a Video

Click Here

dixie chicks not ready to make nice

4:59

The newest video from the **dixie chicks!** "not ready to make nice"

Tags: dixie chicks music, video, old, music, in, music, like

Added: 1 year ago in Category: Entertainment

From: [dixiechicks](#)

Views: 1,362,915

4.8 rating

Dixie Chicks - Wide Open Spaces

3:27

More of the **Chicks** at their best!

Tags: dixiechicks, wide open spaces

Added: 10 months ago in Category: Music

From: [dixie](#)

Views: 765,201

4.8 rating

Dixie Chicks - Not Ready to Make Nice

4:16

Dixie Chicks - Not Ready to Make Nice

Tags: Dixie Chicks

Added: 11 months ago in Category: Music & Videos

From: [dixiechicks](#)

Views: 251,520

4.8 rating

Dixie Chicks Landslide

3:54

Dixie chicks sing landslide live

Tags: dixie chicks, landslide, live

Examples of social search

Browsing content resources

flickr® from YAHOO!


Home The Tour Sign Up Explore


You aren't signed in [Sign In](#) [Help](#)


Search


Jump to:

[Explore](#) / [Tags](#) / [fresco](#) / [clusters](#)


[Italy](#), [Italia](#), [rome](#), [roma](#), [pompeii](#), [vatican](#),
[affresco](#), [chiesa](#), [europa](#), [roman](#)
[See more in this cluster...](#)

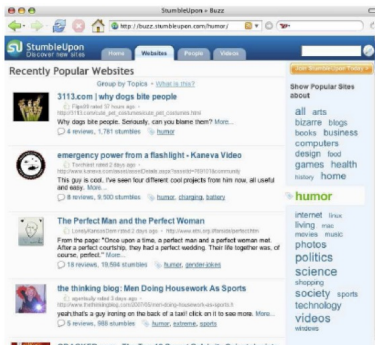

[church](#), [painting](#), [art](#), [ceiling](#), [architecture](#),
[mural](#), [dome](#), [turkey](#), [wall](#), [orthodox](#)
[See more in this cluster...](#)


[fresh](#), [water](#), [green](#), [verde](#), [agua](#), [summer](#),
[acqua](#), [estate](#), [fruit](#), [food](#)
[See more in this cluster...](#)


[florence](#), [firenze](#), [tuscany](#), [duomo](#), [toscana](#),
[cathedral](#)
[See more in this cluster...](#)

Examples of social search

Content recommendation



The screenshot shows a web browser window displaying the StumbleUpon website. The page title is "StumbleUpon - Buzz" and the URL is "http://buzz.stumbleupon.com/humor/". The main content area is titled "Recently Popular Websites" and lists several items:

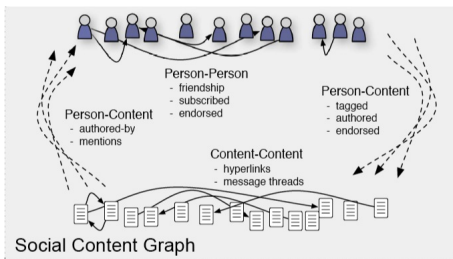
- 3113.com | why dogs bite people**
Why dogs bite people. Seriously, can you blame them? More...
4 reviews, 1,781 stumbles
- emergency power from a flashlight - Kanava Video**
This guy is cool. I've seen four different cool projects from him now. all useful and easy. More...
8 reviews, 9,500 stumbles
- The Perfect Man and the Perfect Woman**
From the page: "Once upon a time, a perfect man and a perfect woman met. After a perfect courtship, they had a perfect wedding. Their life together was, of course, perfect." More...
18 reviews, 19,554 stumbles
- the thinking blog: Men Doing Housework As Sports**
years that's a guy taring on the back of a taxi! click on it to see more. More...
5 reviews, 988 stumbles

On the right side, there is a sidebar titled "Show Popular Sites about" with a list of categories: arts, bizarre, blogs, books, business, computers, design, food, games, health, history, home, humor, internet, linux, living, mac, movies, music, photos, politics, science, shopping, society, sports, technology, videos, and videos.

Ranking search results in a social content graph.

The social content graph

- To capture the interactions between content linking and endorsements from friends, it is natural to treat resource endorsements, friendships links and people endorsements of content uniformly as edges in a **social content graph**.



An integrated approach to relevance computation

- How do we model a random surf on the social content graph?
- First, the system assigns node and edge weights to be used for random surfing to nodes and edges in the social content graph.
- Second, the stationary distribution that a surfer arrives at each node is computed.
- Third, the k nodes with the highest such probability are returned to the user.

Integrated approach to relevance computation

From relevance factors to transition probabilities

- How do we adjust the parameters of the computation to handle the relevance factors?
- Text features: set the node weights proportional to the query relevance of the text associated with a resource to the query terms.
- Timeliness, freshness and popularity: adjust edges and node weights by their recency.
- Incoming links and tags: set the edge weights proportional to relevance of the query to the tag or other text associated with the edge in the social-content graph;
- Social distance: apply a significant fraction of the total node weight to the querier's node and adjust Person-to-Person edge weights according to the strength of the connection.

Integrated approach to relevance computation

Feasibility and performance

- The stationary probability can be calculated using a fix-point algorithm.
- The probabilities are only known at query-time: off-line computation is not possible.
- Possible solution: use random surfing for only a subset of the features, and rely on existing Web techniques for the remainder.

Modular approach to relevance computation

- The integrated approach accounts for social distance in a very fine-grained way.
- However, it lacks modularity, because it cannot exploit the components already developed for Web search.
- Modular approach: consider each factor (or a subset of the factors) in isolation, producing separate rankings that are averaged to get one final score.
- Example: compute query relevance by traditional TF-IDF scoring of the content, and rank resource endorsements via link analysis.

Modular approach to relevance computation

Computing Social Endorsement

- Example: the query is a keyword search and the social endorsement score of a resource is computed as the number of times friends of the querier have tagged the resource with query-matching tags.
- What sort of indexing structure is available for tagging information?
- Which algorithm do we use for query evaluation?
- The most natural approach is to organize data in inverted lists by tag;
- Each entry in the list records a resource identifier and also its list of taggers.

Computing social endorsement

- The score of a resource can be computed as the number of its taggers who are friends of the querier, or as the sum of all the people in its connected component. weighted by social distance.
- Social endorsement score: combining the rankings of different lists;
- The difference from standard rank combination problems is that exactly which users in the list contribute to the score is dependent on the querier.
- Standard algorithms for combining scores rely on the sorting of the inverted lists by static upper-bound scores.

Modular approach to relevance computation

Refining scores by clustering

- The integrated approach is based on the notion of community given by explicit friendship information.

- We may want to use derived notion of affinity between users, creating either links between users or clusters of users based on common behavior.

- How do we incorporate social behavior into processing search queries?
- How do we rank an answer according to its popularity among members of a seeker's networks?

- We model collaborative systems as follows:
- Users can be either **taggers** or **seekers**.
- **Taggers** annotate **items** with one or more **tags**.
- A **query** is composed by a set of tags and is asked by a seeker;
- A linking relation connects seekers and taggers, forming the network associated with each seeker;
- Seekers and taggers can be the same.
- Given a seeker, a networks of taggers, and a query in the form of a set of tags, we wish to return the most relevant items.

Network-aware search

Data model

- We model the social network underlying a collaborative tagging system as a directed graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$
- $\text{Items}(v)$: set of items tagged by user v with any tag;
- $\text{Tagged}(v, i, t)$: tagger v tags item i with tag t .
- $\text{Link}(u, v)$:
 - A large fraction of the items tagged by u , are also tagged by v :

$$\text{Link}(u, v) : |\text{Items}(u) \cup \text{Items}(v)| / |\text{Items}(u)| > th$$

- Alternative definition: v tags a sufficient fraction of the items tagged by u with the same tag as u , i.e.,

$$|\{i | \exists t : \text{Tagged}(u, i, t) \cup \text{Tagged}(v, i, t)\}| / |\{i | \exists t : \text{Tagged}(u, i, t)\}| > th$$

- The projection on the first component of **Link** is the **Seekers** set;
- The projection on the first component of **Tagged** represents the **Taggers** set.
- For a seeker $u \in \mathbf{Seekers}$, **Network**(u) is the set of neighbors of u , i.e.,
 $\mathbf{Network}(u) = \{v \mid \mathbf{Link}(u, v)\}$
- $Items(v, t) = \{i \mid Tagged(v, i, t)\}$ set of items tagged with t by tagger $v \in Taggers$;
- We define the score of an item i for user u w.r.t. a tag t_j as a monotone function of the number of taggers in u 's network who tagged i with tag t_j , i.e.,

$$\mathbf{score}_{t_j}(i, u) = f(|\mathbf{Network}(u) \cap \{v \mid Tagged(v, i, t_j)\}|)$$

- We define the overall query score for a seeker $u \in Seekers$ as a monotone aggregation of the scores for the individual keywords of the query, i.e.,

$$score(i, u) = g(\mathbf{score}_{t_1}(i, u), \dots, \mathbf{score}_{t_n}(i, u))$$

- In the following, $f = count$ and $g = sum$.

Network-aware search

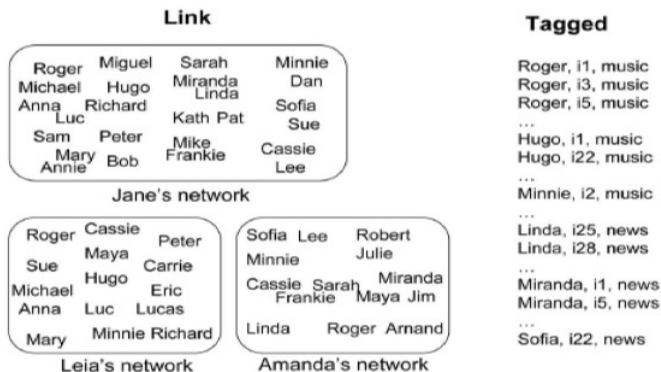


Figure: Network-aware-search: seekers, networks, and tagging actions

- **Problem statement:** Given a query $Q = t_1, \dots, t_n$ issued by user u , and a number k , we want to efficiently determine the top k items, i.e., the k items with the highest overall score.
- What kind of information should we precompute in order that well-known top- k algorithms can be leveraged, and how should we adapt these algorithms to work correctly and efficiently in our setting?
- We organize items in *inverted lists* and study the applicability of typical top- k processing algorithms to our set-up.

Computing exact scores: Exact method

- Typical IR approach: create one inverted list for each keyword;
- Each entry in the list contains a doc-id together with its score for that keyword;
- Storing scores allows to sort entries in the inverted list, therefore enabling top- k pruning;
- In our problem, the score of an item for a tag depends on *who* is asking the query, i.e., the seeker's network.
- Simple adaptation: maintain one inverted list per *tag,seeker* pair and sort the items in each list according to their score for tag and seeker.
- Major drawback: prohibitive storage requirements.

Computing exact scores: Example

- Each item is replicated along with its exact score in each $(tag, seeker)$ network.
- For each tag, the exact method stores as many inverted lists as there are networks, and there could be one per seeker in the worst case.
- Consider a collaborative tagging system with $100K$ users, $1M$ items, and $1K$ distinct tags.
- Suppose that on average each item receives 20 tags that are given by 5% of the users.
- Index size: $100K$ seekers \times $1M$ items \times 20 tags \times 5% \times size of each entry.
- Assuming 10 bytes per entry, the size of the index is 1TB.
- Actual numbers in real systems may be much higher.

Top-K processing with exact scores: NRA (No Random Access)

- The inverted list for each query keyword is assumed to be sorted on the exact score of items.
- The algorithm maintains a heap containing the current candidates for top k.
- The inverted lists are scanned sequentially in parallel;
- When a new item is found, it is added to the heap along with its partial exact score;
- If the item was seen before, its score in the heap entry is updated;
- For every heap entry, maintain a worst-case score and a best-case score.

Top-K processing with exact scores: NRA (No Random Access)

- The worst-case score is based on the assumption that the item does not appear in the lists where it is so far unseen;
- The best-case score is based on the assumption that the score for an item in a list where it is unseen is equal to the bottom score of the heap;
- Items in the heap are sorted according to the worst-case score;
- The algorithm stops when none of the items outside the top k items found so far has a best score higher than the worst score of the k-th item in the buffer.
- In our problem, the algorithm can be adapted by picking the lists that correspond to the current seeker and query keywords and aggregating them as described above.

Top-K processing with exact scores: TA (Threshold algorithm)

- The inverted lists are sorted on score, but random access is assumed in addition to sequential access.
- The algorithm accesses items in the various lists sequentially and in parallel.
- It maintains a heap, where items are kept sorted on their complete score.
- When a new item is seen in a list under sequential access, its scores from other lists are obtained by random access.

Top-K processing with exact scores: TA (Threshold algorithm)

- If the overall score is higher than the k -th entry in the heap, they are swapped, otherwise the new item is discarded.
- The bottom score is maintained for every list and used to compute a threshold;
- The algorithm stops when the k -th highest heap entry is greater or equal than the threshold.
- Output: top- k lists in the buffer, including items and their scores.

Computing score upper bounds

- Maintain entries of the form (item, itemTaggers), where itemTaggers are all the taggers who tagged the item with the tag.
- each item is stored at most once in the inverted list for a given tag.
- Which score do we store with each entry?
- The maximum score that an item could have for a tag across all the possible seekers.
- Global upper bound strategy: the upper bound score is the max score that an item could have for a tag.

Score upper bounds: example

Global Upper-Bound strategy

<i>item</i>	<i>taggers</i>	<i>upper-bound</i>
i8	Miguel,...	73
i19	Kath, ...	65
i7	Sam, ...	62
i5	Miguel, ...	53
i24	Peter, ...	45
i9	Jane, ...	44
i15	Mary, ...	43
i39	Miguel, ...	22
i27	Kath, ...	17
i21	Mary, ...	16
...		

all seekers

Exact strategy

<i>item</i>	<i>exact score</i>
i8	73
i19	65
i7	62
i15	40
i24	39
i39	18
i21	16
i27	16
...	

seeker Jane

<i>item</i>	<i>exact score</i>
i5	53
i9	36
i19	30
i39	15
i24	14
i27	10
i21	10
i7	5
...	

seeker Leia

<i>item</i>	<i>exact score</i>
i8	25
i19	23
i39	22
i24	20
i7	19
i5	15
i27	12
i21	10
...	

seeker Amanda

Figure: Inverted lists organization for the tag *music*

Top-k processing with score upper bounds

Basic assumptions

- Assume a function that takes a seeker and a pair (item,itemTaggers) and compute from the inverted list the number of item taggers who are friends of the seeker.

- Assume a function that retrieves the taggers of item i for tag t_j and computes the number of friends of user u who tagged with t_j .

Top-k processing with score upper bounds

NRA generalization

Algorithm 1 Bounds-Based NRA Algorithm (gNRA)

Require: seeker u , Query Q ;

- 1: Open inverted lists IL_t for each keyword $t \in Q$;
 - 2: **while** $\text{worstcase}(k\text{th heap item}) \leq \text{max}\{\text{BestcaseUnseen}, \text{max}\{\text{bestcase}(j) \mid j \in \text{heap} - \text{top-}k\}\}$
do
 - 3: Get next list IL_t using round-robin;
 - 4: Get entry $e = (i, \text{ub}, \text{itemTaggers})$ in IL_t ;
 - 5: Update the bottom bound of IL_t ;
 - 6: Compute partial exact score of i for t using itemTaggers ;
 - 7: If i is not in heap add it, otherwise update its partial exact score;
 - 8: Update best-case scores of items in heap, and re-order heap;
 - 9: $\text{BestcaseUnseen} = \text{sum of bottom bounds over all lists}$;
 - 10: **end while**
 - 11: Return top- k set of items from heap.
-

Top-k processing with score upper bounds

NRA generalization

- Access sequentially the inverted lists of each query keyword;
- When a query term is encountered in a list:
 - record the upper bound of the item
 - compute the exact score of the item for that tag
- Scores of items in the heap are partial exact scores (worst case NRA);
- The set of bottom bounds of the list is used to compute best-case scores of the items.
- The heap is kept sorted on worst-case scores, with ties broken on the best-case score.
- Stopping condition: none of the items outside the top-k in the heap has a best-case score that is more than the worst-case score of the k-th item in the heap

Top-k processing with score upper bounds

NRA generalization

Several optimizations are possible:

- No need to update scores of items in the heap whose best score is below the worst score of the k -th highest heap item;
- No need to reorder when the lower bounds are updated;
- Check top- k candidates by comparing their new upper bounds against the worst-case score of the current k -th item;

Top-K processing with score upper bounds

TA generalization

Algorithm 2 Bounds-Based TA Algorithm (gTA)

Require: seeker u , Query Q

- 1: Open inverted lists IL_t for each keyword $t \in Q$;
 - 2: **while** $\text{score}(k\text{th heap item}) \leq$
sum of bottom bounds over all lists **do**
 - 3: get next list IL_t using round-robin;
 - 4: Let $e = (i, \text{ub}, \text{itemTaggers})$ be the next entry in IL_t ;
 - 5: Update the bottom bound of IL_t ;
 - 6: **if** i not in current top- k **then**
 - 7: Use local aggregation to get exact score of i in IL_t using
itemTaggers;
 - 8: Use `computeExactScore` to get exact score of i in other
lists;
 - 9: **if** i 's overall score $>$ k th score in heap **then**
 - 10: Swap k th item with i ; keep top- k heap sorted;
 - 11: **end if**
 - 12: **end if**
 - 13: **end while**
 - 14: Output the heap as is.
-

Top-K processing with score upper bounds

TA generalization

- Given a query from a seeker, all the relevant inverted lists are identified and accessed sequentially.
- When an entry is seen for the first time under sequential access in a list, we compute its exact score in that list and in the other lists as well.
- Remember the bottom bound seen for each list.
- The threshold is the sum of the bottom bounds over all lists
- The algorithm stops whenever the score of the k-th item in the heap is no less than the threshold.

Top-K processing with score upper bounds

TA generalization: Possible optimizations

- Both variants of Global Upper Bound differ from exact in that the former needs to compute upper-bound scores for a seeker and tag at query time.
- Both variants are instance-optimal over all the algorithms that use the same upper-bound based storage.
- The accuracy of upper bounds in the inverted lists is clearly the key factor in the efficiency of the top-k pruning.
- The finer the upper bound, the faster an item can be pruned.
- Further optimizations are needed

Clustering-based approach

- Goal: reduce the distance between exact scores and upper bounds.
- Core idea: cluster users into groups and compute score upper bounds within each group.
- Intuition: if a cluster contains users whose behavior is similar, then the exact scores of users in the clusters would be very similar to their upper bounds.
- Which users should the algorithm cluster to achieve this goal?

Clustering seekers

- Given any clustering of seekers, we form an inverted list for every tag and for every cluster;
- The score of an item is the maximum score over all the seekers in the cluster;
- Query processing: find the cluster containing the user and then perform aggregation over the inverted lists to be considered;
- Global-Upper-Bound is a special case of Clustering seekers where all the seekers fall into the same cluster and the same cluster is used for all tags.

Clustering seekers

inverted list for tag “*music*”

<i>item</i>	<i>exact score</i>	<i>item</i>	<i>exact score</i>
i8	73	i7	53
i7	62	i9	42
i19	61	i19	30
i5	53	i8	25
i24	43	i24	22
i9	40	i39	22
i39	21	i15	19
i21	16	i21	17
...		...	

cluster C1
{Jane, Leia,...}

cluster C2
{Amanda,...}

inverted list for tag “*news*”

<i>item</i>	<i>exact score</i>	<i>item</i>	<i>exact score</i>
i3	99	i5	23
i2	87	i6	21
i21	73	i17	19
i5	40	i8	17
i19	35	i15	11
i8	32	i9	5
i24	18	i21	3
i48	11	i18	2
...		...	

cluster C1
{Amanda, Leia,...}

cluster C2
{Jane,...}

- Single-keyword queries: Exact is optimal over all algorithms that use Cluster-Seekers, for any clustering;
- Multi-keyword queries:
- An ideal clustering should take into account both the scores and the ordering.

How do we cluster seekers?

- Ideally, we would find a clustering that is optimal for the running time of one of our algorithms;
- We could look at the worst-case running time of all users;
- It can be shown that finding a clustering that satisfies this criterion is NP-hard, even for the trivial cases;
- Similar results are achieved when considering the average-case computing time for a cluster;

How do we cluster seekers?

- Thus, we must rely on heuristic methods to find clusters of seekers;
- Natural approach: consider the overlap of the seekers' networks;
- Intuition: given two seekers, the higher the number of common taggers in their networks, the higher the chance of the score of an item to be similar for those two networks;
- Idea: Compute per-tag network overlap between seekers; e.g., build a graph where the nodes are seekers and two nodes are connected by an edge iff the sets of users in the networks of the two users who tagged an item with the same tag is above a given threshold; then apply graph-clustering algorithms

Clustering taggers

- Alternative approach: organize taggers into groups which reflect overlap in their tagging behavior;
- Cluster-Taggers strategy: for each tag, partition the taggers into clusters;
- Again, form lists on a per-tag, per-cluster basis;
- The score assigned to an item is the max n. of taggers in the cluster who are linked to a seeker and tagged the item with the same tag;
- Query processing: find the clusters of the taggers in the network of the seeker, then aggregate over the inverted lists associated with the (tag,cluster) pairs;
- Members of a seeker's network may fall into multiple clusters for the same tag, thus requiring to process more lists for each tag;
- Build a graph where the nodes are taggers and edge exists between any two nodes iff the number of items that the two taggers tagged with the same tag is higher than a given threshold.

Differences between the two clustering approaches:

- Cluster-Seekers: cluster seekers based on network overlap;
- Cluster-Taggers: cluster taggers based on overlap in tagging behavior;
- At query time, Cluster-Seekers identifies an inverted list per (tag,seeker) pair since a seeker always falls into a single cluster for a tag;
- Cluster-Taggers may need to access multiple inverted lists per (tag, seeker) pair, given that a seeker may have multiple taggers in her network, and these taggers may fall into different clusters;
- Cluster-Seekers replicates tagging actions over multiple inverted lists;

Differences between the two clustering approaches:

- Cluster-Taggers does not introduce a significant penalty in terms of space.
- Processing time: Cluster-Seekers benefits all seekers; Cluster-Taggers can hinder seekers that are associated with many tagger clusters and hence many inverted lists;
- Maintenance under updates: Cluster-Seekers requires multiple exact score computations and as new tagging events occur;
- Cluster-Taggers only requires a single exact score computation and a single update for each tag.

- P. Heymann, G. Koutrika, and H. Garcia-Molina.
Can social bookmarking improve web search?
In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 195–206, New York, NY, USA, 2008. ACM.
- C. Marlow, M. Naaman, D. Boyd, and M. Davis.
Ht06, tagging paper, taxonomy, flickr, academic article, to read.
In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 31–40, New York, NY, USA, 2006. ACM.
- S. A. Yahia, M. Benedikt, and P. Bohannon.
Challenges in searching online communities.
IEEE Data Eng. Bull., 30:2007, 2007.
- S. A. Yahia, M. Benedikt, L. V. S. Lakshmanan, and J. Stoyanovich.
Efficient network aware search in collaborative tagging sites.
Proc. VLDB Endow., 1(1):710–721, 2008.