Taagle: Efficient, Personalized Search in Collaborative Tagging Networks

Silviu Maniu　　Bogdan Cautis
Institut Mines-Télécom - Télécom ParisTech; CNRS LTCI
Paris, France
first.last@telecom-paristech.fr

ABSTRACT

We demonstrate the Taagle system for top-k retrieval in social tagging systems (also known as *folksonomies*). The general setting is the following: users form a weighted social network, which may reflect friendship, similarity, or trust; items from a public pool of items (e.g., URLs, blogs, photos, documents) are tagged by users with keywords; users search for the top-k items having certain tags. Going beyond a classic search paradigm where data is decoupled from the users querying it, users can now act both as producers and seekers of information. Hence finding the most relevant items in response to a query should be done in a network-aware manner: items tagged by users who are closer (more similar) to the seeker should be given more weight than items tagged by distant users.

We illustrate with Taagle novel algorithms and a general approach that has the potential to scale to current applications, in an online context where the social network, the tagging data and even the seekers' search ingredients can change at any moment. We also illustrate possible design choices for providing users a fully-personalized and customizable search interface. By this interface, they can calibrate how social proximity is computed (for example, with respect to similarity in tagging actions), how much weight the social score of tagging actions should have in the result build-up, or the criteria by which the user network should be explored. In order to further reduce running time, seekers are given the possibility to chose between exact or approximate answers, and can benefit from cached results of previous queries (materialized views).

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search Process*

Keywords

social applications, social search, threshold algorithms

1. INTRODUCTION

Unprecedented volumes of data are now at everyone's fingertips on the Web. A new dynamics to this development has been brought by the *social Web*, applications that are centered around users, their relationships and their data. User-generated content is indeed becoming a significant, highly qualitative portion of the Web. This

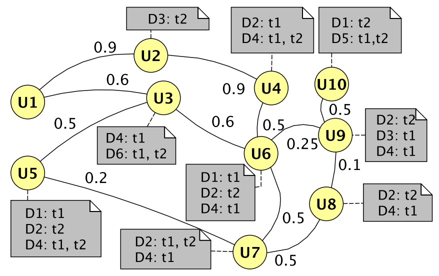


Figure 1: A folksonomy and its social network.

calls for adapted retrieval techniques. going beyond a classic search paradigm where data is decoupled from the users querying it.

An important category of social applications are the *collaborative tagging sites*, with popular examples including Del.icio.us, StumbleUpon or Flickr. Their general setting is the following:

- users form a *social network*, which may reflect proximity, similarity, friendship, closeness, trust, etc,
- items (e.g., document, URLs, photos, etc) are *tagged* by users with keywords, for purposes such as description and classification, sociality, or to facilitate later retrieval,
- users *search* for items having certain keywords (i.e., tags) or they are *recommended* items.

In this context, finding the most relevant items that are tagged by some keywords should be done in a *network-aware* manner, following the intuition that items tagged by users who are closer to the seeker should be given more weight, as supported also by studies on the relevance of answers [13].

We associate with the notion of social network a rather general interpretation, as a user graph whose edges are labeled by *social scores*, which give a measure of the proximity or similarity between two users. [1] These relationships are then exploitable in searches, as indicators for how much weight one's tagging actions should have in the result build-up. Following a natural interpretation that user links (e.g., similarity or trust) are (at least to some extent) transitive, even items and tags from users who are only indirectly (or implicitly) connected to the seeker should be relevant for a query.

We demonstrate the Taagle approach for top-k retrieval in collaborative tagging systems. It relies on novel techniques, that have the potential to scale to current applications on the Web. (The most popular ones have user bases of the order of millions and huge repositories of data.)

EXAMPLE 1. *Consider the collaborative tagging configuration of Figure 1. Users have associated lists of tagged documents and*

[1]Even in applications where an explicit social network does not exist or is not exploitable, one may use the tagging history to build a network based on similarity in tagging and items of interest.

they are interconnected by social links. Each link is labeled by its (social) score, assumed to be in the $[0, 1]$ interval. Let us consider user $u1$ in the role of the seeker, looking for the top-2 documents that are tagged with both $t1$ and $t2$. Looking at $u1$'s immediate neighbors and their respective documents, intuitively, $D3$ should have a higher score than $D4$, given that the former is tagged by a more relevant user ($u1$, whose social score is the maximal one). If we expand the search to the entire social graph, the score of $D4$ may however benefit from the fact that other users, such as $u5$ or even $u8$, also tagged it with $t1$ or $t2$. Furthermore, documents such as $D2$ and $D1$ may also be relevant for the top-2 result, even though they were tagged only by users who are indirectly linked to $u1$. Under certain assumptions to be clarified shortly, the top-2 documents for $u1$'s query could be, in descending score order, $D4$ and $D2$. In particular, this would be due to the fact that $u4$'s (implicit) proximity with respect to seeker $u1$ can be given by $0.81 = max(0.9 \times 0.9, 0.6 \times 0.6 \times 0.5)$.

Taagle is designed with a focus on three key aspects:

1. *Efficiency:* to enable good running time over large networks and collections of items, we proposed in [12] an algorithm that is instance optimal (over a large and important class of algorithms). Moreover, it can access on-the-fly the closest users for a given seeker, for a large family of functions for proximity computation (including the most natural ones). For further efficiency, we also provide users with techniques for approximate results, while striving to maintaining high accuracy. In addition, we exploit in Taagle cached results of previous queries (materialized views).
2. *Applicability:* the data structures and indexes it uses should be applicable at the scale of current applications and beyond, without requiring a prohibitively large amount of memory; also, these should be able to cope with very frequent updates on the social network and tagging of items.
3. *Customization–personalization:* seekers are given complete control over choices w.r.t. the scoring and ranking model, the computation of proximity values and how the social network is explored, the kind of results (exact or approximate), etc.

Main related work. [1] is the first to consider the problem of network-aware search in collaborative tagging, though by a simplified flavor. The authors consider an extension to classic top-k retrieval in which social proximity is seen as a binary function. In [13], the network-aware retrieval problem for collaborative tagging is considered under the general interpretation, positing that even users who are only implicitly connected to the seeker can be relevant for the top-k result. The main drawbacks of [13] are scalability and applicability. They require precomputing a weighted transitive closure over the entire network for proximity, which has a very high cost in terms of space and computation in even moderate-size social networks. Also, keeping these proximity lists up-to-date – especially when they reflect tagging similarity – would simply be unfeasible in real-world settings, which are highly dynamic. The techniques of [13] have been demonstrated in a system in [6]. The topic of search in a social setting has received increased attention lately. Studies and models of personalization of social tagging sites can be found in [9, 7, 14]. Other studies have found that including social knowledge in scoring models can improve search and recommendation algorithms. In [5], personalization based on a similarity network is shown to outperform other personalization approaches and the non-personalized social search. A study on a last.fm dataset in [11] has found that incorporating social knowledge in a graph model system improves the retrieval recall of music track recommendation algorithms. An architecture for social data management is given in [2, 3], along with a framework for information discovery and presentation in social content sites. Another approach to rank resources in social tagging environments is CubeLSI [4], which uses a vector space model and extends LSI to include taggers in the feature space of resources, in order to better match queries to documents. FolkRank [10] proposes a ranking model in folksonomies, for recommendation and search, by an adaptation of PageRank on the graph of users, tags, and resources.

2. MODEL AND ALGORITHMS OVERVIEW

We briefly describe in this section the top-k social search model and the algorithm (TOPKS) on which Taagle relies. For a more detailed description of TOPKS, we refer the reader to [12].

We consider a social setting in which we have a set of items (could be text documents, URLs, photos, etc) $\mathcal{I} = \{i_1, \ldots, i_m\}$, each tagged with one or more distinctive tags from a dictionary of tags $\mathcal{T} = \{t_1, t_2, \ldots, t_l\}$ by one or more users from $\mathcal{U} = \{u_1, \ldots, u_n\}$. Users form an undirected weighted graph $G = (\mathcal{U}, E, \sigma)$ called the *social network*. In G, each node is a user and σ is a function that associates to each edge $e = (u_1, u_2)$ a value in $(0, 1]$, called *the proximity* (or social) score between u_1 and u_2.

Given a seeker user s, a keyword query $Q = (t_1, ..., t_r)$ (a set of r distinct tags) and an integer value k, the top-k retrieval problem is to compute the list of the k items having the highest scores with respect to the seeker and query.

We first model for a user, item and tag triple (u, i, t) the *score* of i w.r.t. seeker u and tag t. This is denoted $score(i \mid u, t)$. Generally,

$$score(i \mid u, t) = h(fr(i \mid u, t))$$

where $fr(i \mid u, t)$ is the *overall term frequency* of i w.r.t. seeker u and tag t, h is a positive monotone function (e.g., tf-idf or BM25).

The overall term frequency function $fr(i \mid u, t)$ is defined as a combination of a network-dependent component and a document-dependent one, with parameter $\alpha \in [0, 1]$, as follows:[2]

$$fr(i \mid u, t) = \alpha \times tf(t, i) + (1 - \alpha) \times sf(i \mid u, t).$$

The former component, $tf(t, i)$, is the term frequency of t in i, i.e., the number of times i was tagged with t. The latter component stands for social frequency, a measure that depends on the seeker.

With each user bringing her own weight (proximity) to the score of an item, we define the measure of social frequency as follows:

$$sf(i \mid u, t) = \sum_{v \in \{v \mid Tagged(v,i,t))\}} \sigma(u, v).$$

Then, given a query q as a set of tags $(t_1, \ldots, t_r)$, the overall score of i for seeker u and query q,

$$score(i \mid u, q) = \sum_{t_j \in Q} score(i \mid u, t_j).$$

The above scoring model takes into account so far only the users directly connected to the seeker. But this can be extended to deal also with users that are indirectly connected to the seeker (friends-of-friends and beyond), by inferring from the explicit σ values the *implicit ones*, for any pair of users connected by a path in the network. This leads to an overall item scoring scheme that depends on the entire network instead of only the seeker's vicinity.

A natural candidate for computing implicit proximity is to multiply the weights on a given path between u and v and then choose the maximum value over all the possible paths ([13, 12]). We can aggregate the weights on a path $p = (u_1, \ldots, u_l)$ (with a slight abuse of notation) as $\sigma(p) = \prod_i \sigma(u_i, u_{i+1})$.

[2]Note that the extreme case of $\alpha = 1$ leads to a scoring model in which the social network is ignored (the classic top-k setting), while $\alpha = 0$ yields an "exclusively social" scoring model.

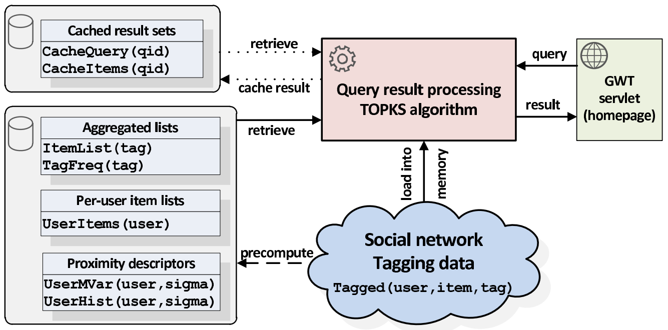


Figure 2: General system architecture.

A possible drawback of multiplication for proximity aggregation is that values may decrease quite rapidly. An alternative that avoids this could be obtained by using minimum over a path instead, as $\sigma^+(p) = min\{\sigma(u_i, u_{i+1})\}$. Under this σ candidate, the values with respect to seeker $u1$ in Figure 1 would be the following: $\{u2 : 0.9, u4 : 0.9, u3 : 0.6, u6 : 0.6, u5 : 0.5, u7 : 0.5, u8 : 0.5, u9 : 0.25, u10 : 0.25\}$.

Another natural candidate for σ relies on a "drop parameter" $\lambda \geq 1$, controlling the speed of the decrease of proximity values, as $\sigma(p) = \lambda^{-\sum_i \frac{1}{\sigma(u_i, u_{i+1})}}$. Under this candidate for σ, for $\lambda = 2$, the rounded values w.r.t seeker $u1$ in Figure 1 would be $\{u2 : 0.81, u3 : 0.36, u4 : 0.21, u6 : 0.09, u5 : 0.07, u7 : 0.03, u9 : 0.018, u8 : 0.018, u10 : 0.011\}$.

We can then define σ for any pair of user (s, u) who are connected in the network by taking the maximal weight over all their connecting paths. More formally, we define $\sigma(s, u)$ as

$$\sigma(s, u) = max_p\{\sigma(p) \mid s \overset{p}{\rightsquigarrow} u\}.$$

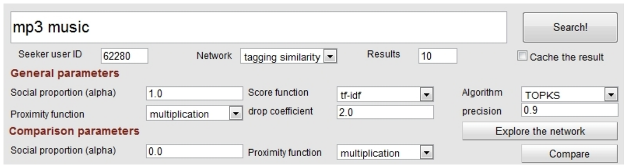
The key common feature of the candidate functions previously discussed is that they are monotonically decreasing over any path they are applied to, when σ draws values from the interval $[0, 1]$; any aggregation function having this property could be used instead.

The TOPKS algorithm. We proposed in [12] and demonstrate here the TOPKS algorithm, in both an exact and approximate version. It belongs to the class of early-termination threshold algorithms (as [8]'s NRA and TA, or [13]'s CONTEXTMERGE).

In short, to find the top-k items for a given query, threshold algorithms scan sequentially (for a given tag) and in parallel (for a given query), relevant lists that are ordered descending by score. During a run, they maintain a list of candidate items already encountered D, ordered descending by their minimal (or guaranteed) scores. At certain intervals, they compare the minimal score (MINSCORE) of the kth item $D[k]$ to the maximal score of items in D outside the top k (the threshold), MAXSCORE, and the maximal score of items not yet encountered, MAXSCOREUNSEEN. When both these maximal scores are not greater than the minimal score of $D[k]$, the run stops, returning the items $D[1], \ldots, D[k]$ as the top-k.

In our social setting, a similar algorithm needs to scan (i) the per-tag inverted lists (that are network-agnostic), (ii) the seeker's proximity list, in descending order, i.e., visiting the next closest unvisited user at each step, and (iii) the per-visited user tagged items (user lists). Our algorithm TOPKS and [13]'s CONTEXTMERGE use such an approach. However, the latter relies on precomputed and exhaustive proximity lists for each possible seeker (hence a complete pre-computed transitive closure).

TOPKS exploits the descending monotonicity property of the proximity aggregation functions described previously to built the proximity vector of the seeker on-the-fly and on demand. This is done by a generalization of Djisksta's algorithm, and has three main advantages. First, we can iterate over the relevant users in more efficient manner, since a typical network can easily fit in main-memory; this can spare the potentially huge disk volumes required by [13]'s algorithm while also having the potential to run faster. Second, changes in the social network can be handled virtually in transparent manner; moreover, when the social network depends on the tagging history, we can keep it up-to-date and, by it, all the proximity values at any given moment, with little overhead. Third, in the Taagle implementation, we can give seekers the possibly to choose and calibrate at query time the proximity model.


mp3 music

Search!

Seeker user ID 62280 Network tagging similarity Results 10 ☐ Cache the result

General parameters

Social proportion (alpha) 1.0 Score function tf-idf Algorithm TOPKS

Proximity function multiplication drop coefficient 2.0 precision 0.9

Explore the network

Comparison parameters

Social proportion (alpha) 0.0 Proximity function multiplication

Compare

Figure 3: Main window and options.

To achieve *instance optimality* in the exclusively social case ($\alpha = 0$), TOPKS dynamically "consumes" (or prunes) the top item in an inverted list when it is first encountered in a user list. In short, this allows tighter evaluations for maximal scores. For arbitrary values of α, the algorithm chooses between the network-agnostic branch (the per-tag inverted lists) and the network exploration (and user lists), based on heuristics validated by extensive experiments.

Regarding the approximate version of TOPKS, score estimates can be further tightened if one has access to high-level yet easily maintainable descriptions of the per-seeker proximity lists (mean and variance, as in [12], or histograms in Taagle). One can then efficiently use these descriptions to derive tighter bounds, controlled by a probabilistic parameter, for both maximal and minimal score estimations. We show in [12] that this approach achieves significant savings in execution time, while ensuring high accuracy.

3. EXPLOITING CACHED RESULTS

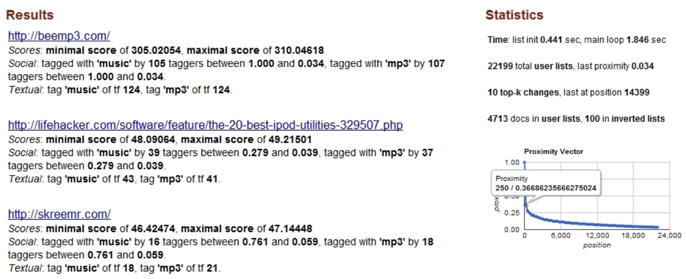
For further efficiency, in the online context of Taagle , users can opt to use materialized results of previous queries in the network.

These precomputed results have an intentional definition, the triple query-seeker-alpha parameter $V(Q^{(V)}, u^{(V)}, \alpha^{(V)})$. They consist of items for which an upper and lower bound on the exact score is known, $(i^{(V)}, lb_i^{(V)}, ub_i^{(V)})$.

We incorporate in Taagle the usage of information present in these views, to refine score ranges for pertinent candidate items during the run of the search, for the current triple of seeker s, query Q, and parameter α. The following factors need to be taken into account when using precomputed results (the views). When one does not have access to precomputed results for the entire input query Q, but instead only for subsets of it or for overlapping queries, one needs to combine the views of one user u into one global view, containing only items having a valid score ranges for the query Q. These bounds are obtained by solving linear programming instances. Moreover, the relationship between the view's $\alpha^{(V)}$ value and the input α value, as the proximity of the seeker s to the view holder u, $\sigma(s, u)$, need to be incorporated in the final score ranges.

A subroutine for handling views is added to TOPKS, whose role is to analyze and exploit these precomputed results. Each time the main algorithm visits a user u, for which there are precomputed results, this subroutine will do the following:

1. decide whether u's views may lead to early termination. If this is the case, select the most useful ones needed to estimate bounds for item scores w.r.t query Q (this depends on the values $\alpha^{(V)}$ of views and the queries $Q^{(V)}$).
2. if (1), compute refined item score ranges, for a selection of items, via linear programming (either exact or approximate).


Results

http://beemp3.com/
Scores: **minimal score** of **305.02054**, **maximal score** of **310.04618**
Social: tagged with **'music'** by **105** taggers between **1.000** and **0.034**, tagged with **'mp3'** by **107** taggers between **1.000** and **0.034**.
Textual: tag **'music'** of tf **124**, tag **'mp3'** of tf **124**.

http://lifehacker.com/software/feature/the-20-best-ipod-utilities-329507.php
Scores: **minimal score** of **48.09064**, **maximal score** of **49.21501**
Social: tagged with **'music'** by **39** taggers between **0.279** and **0.039**, tagged with **'mp3'** by **37** taggers between **0.279** and **0.039**.
Textual: tag **'music'** of tf **43**, tag **'mp3'** of tf **41**.

http://skreemr.com/
Scores: **minimal score** of **46.42474**, **maximal score** of **47.14448**
Social: tagged with **'music'** by **16** taggers between **0.761** and **0.059**, tagged with **'mp3'** by **18** taggers between **0.761** and **0.059**.
Textual: tag **'music'** of tf **18**, tag **'mp3'** of tf **21**.

Statistics

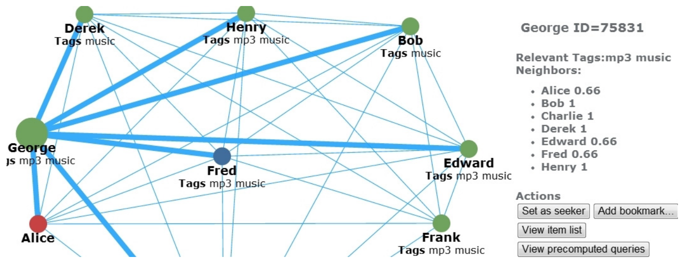
Time: list init **0.441** sec, main loop **1.846** sec

22199 total **user lists**, last proximity **0.034**

10 top-k changes, last at position **14399**

4713 docs in **user lists**, **100** in **inverted lists**

Figure 4: Results window.



George ID=75831

Relevant Tags:mp3 music
Neighbors:

- Alice 0.66
- Bob 1
- Charlie 1
- Derek 1
- Edward 0.66
- Fred 0.66
- Henry 1

Figure 5: Exploring the network and per-user options.

4. ARCHITECTURE AND DEMO SCENARIO

Figure 2 gives an overview of the general architecture of Taagle. The entry point of the application is a GWT servlet via which users formulate top-k queries, using the various options and parameters available. The queries and corresponding parameters are then sent asynchronously, via RPC calls, to the server-side application, which handles the top-k processing. The server-side application is an implementation of the TOPKS algorithm and its variants (see [12] for a detailed description). The results are sent back to the client servlet, which displays them along with relevant metadata (e.g., statistics).

TOPKS uses data from precomputed tables and handles the on-the-fly computation of social proximities. At initialization, it loads the social network into main-memory. The system uses pre-computed projections of the `Tagged` relation: the per-tag inverted lists in `ItemList`, total tag frequencies in `TagFreq` (used for idf values), per-user item lists; it also uses per-user materialized views: the description in `CacheQuery` and the resulting items with their score ranges in `CacheItems`. A per-user item list is only accessed when the algorithm visits the respective user. Finally, the TOPKS approximate version uses a high-level description of each per-user proximity list, stored in `UserMVar` (mean and variance) and `UserHist` (histograms).

We demonstrate an implementation of this architecture, enriched with tools that assist visitors in understanding the principles of the system and the impact of its various parameters. The system uses the tagging data from a subset of the Delicious social community, including about 80000 users and 600000 bookmarked items.

After "logging in", Taagle visitors identify with one of the members of the network, and are able to formulate queries from that perspective (as the seeker). They are able to fully customize their search experience using the interface in Figure 3, having access to (i) a selection of ways to compute similarity measures (used for proximity computations) such as tag, item or item-tag similarity, (ii) various proximity aggregation functions (path multiplication, minimum, the parameterized function with drop parameter λ, or enter their own), (iii) tools for comparing the results and the behavior of the various algorithms implemented in the system (TOPKS, the exact algorithm; TOPKS/MVAR and TOPKS/HIST, approximate algorithms using high-level descriptions of proximities, controlled by a precision parameter; TOPKS/VIEWS, exact algorithm that may exploit precomputed results of other seekers).

The results window, for which a capture is given in Figure 4, will present the top-k items, together with an explanation of the item scores and algorithm statistics: running times, number of users and documents processed, threshold values and termination parameters, interactive plots of the social proximities that were generated and accessed during execution.

For further interaction with the system, visitors will be able to explore and modify part of the collaborative tagging network, and investigate the impact of their changes on query results. Figure 5 presents the exploration interface, where nodes (representing users in the dataset) are color-coded depending on their relationship to the query and the seeker. In the figure, the blue node is the current seeker, the green nodes are users who have items tagged with at least one query tag and were processed for the query (i.e., contributed to candidate item scores), and red nodes are those that were not relevant for the run. The visitor can then select any of these nodes and require information about them: her neighborhood with the respective proximities, the relevant tags used and the possibly relevant views she holds. Several contextual actions are then available to visitors. They can for example set the selected node as the new seeker, as they can tag an item (new or existing) using any list of tags on behalf of the selected user. This helps the visitor notice two possible effects: (i) depending on the selected user, the added (or modified) item may become relevant for the requested top-k, and (ii) the similarity of the selected user relative to other users in the system may change, and this change will be handle transparently and without overhead in future searches.

Taagle visitors will also be able to cache top-k results, and use them in later queries by various seekers. A view exploration screen (omitted here), where we indicate users holding views, will enable visitors to see which users in an immediate vicinity hold some that may be relevant for the current query.

Acknowledgements. The described work was funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement number 270239 (ARCOMEM).

5. REFERENCES

[1] S. Amer-Yahia, M. Benedikt, L. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. In *VLDB*, 2008.

[2] S. Amer-Yahia, J. Huang, and C. Yu. Building community-centric information exploration applications on social content sites. In *SIGMOD*, 2009.

[3] S. Amer-Yahia, L. Lakshmanan, and C. Yu. Socialscope: Enabling information discovery on social content sites. In *CIDR*, 2009.

[4] B. Bi, S. Lee, B. Kao, and R. Cheng. An effective and efficient method for searching resources in social tagging systems. In *ICDE*, 2011.

[5] D. Carmel, N. Zwerdling, I. Guy, S. Ofek-Koifman, N. Har'el, I. Ronen, E. Uziel, S. Yogev, and S. Chernov. Personalized social search based on the user's social network. In *CIKM*, 2009.

[6] T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J. X. Parreira, R. Schenkel, and G. Weikum. Making sense: socially enhanced search and exploration. In *VLDB*, 2008.

[7] Z. Dou, R. Song, and J. Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW07*.

[8] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

[9] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search? In *WSDM*, 2008.

[10] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *ESWC*, pages 111–114, 2006.

[11] I. Konstas, V. Stathopoulos, and M. Jose. On social networks and collaborative recommendation. In *SIGIR*, 2009.

[12] S. Maniu, B. Cautis, and T. Abdessalem. Efficient top-k retrieval in online social tagging networks. `http://arxiv.org/abs/1104.1605`.

[13] R. Schenkel, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J. X. Parreira, and G. Weikum. Efficient top-k querying over social-tagging networks. In *SIGIR*, 2008.

[14] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu. Exploring folksonomy for personalized search. In *SIGIR*, 2008.