Social Search and Discovery using a Unified Approach

Einat Amitay, David Carmel, Nadav Harel, Shila Ofek-Koifman,
Aya Soffer, Sivan Yogev, Nadav Golbandi
IBM Haifa Research Lab
Haifa 31905 Israel
{einat,carmel,nyh,shila,ayas,sivany}@il.ibm.com, nadav.golbandi@gmail.com

ABSTRACT

This research explores new ways for augmenting search and discovery of relations between Web 2.0 entities using multiple types and sources of social information. Our goal is to allow searching for all object types such as documents, persons and tags, while also retrieving related objects of all types. To realize this goal, we implemented a social-search engine using a unified approach. In this approach, the search space is expanded to represent heterogeneous information objects that are interrelated by several relation types. We address a novel solution based on multifaceted search which provides an efficient update mechanism for relations between objects, as well as efficient search over the heterogeneous data. We describe a social search engine positioned within a large enterprise, applied over social data gathered from several Web 2.0 applications. We conducted a large user study with over 600 people to evaluate the contribution of social data for search. Our results demonstrate the high precision of social search results and confirm the strong relationship of users and tags to the topics they were retrieved for.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Social Search, multifaceted Search, enterprise search

1. INTRODUCTION

Recent Web 2.0 applications such as web logs (blogs), collaborative bookmarking systems, and social networks, introduce new entities and relations, in addition to regular Web pages. These new entities and relations may prove valuable in enhancing the search experience as potential search results or by influencing ranking algorithms. In recent surveys of activities of the online user population, 28% of online Americans said they have tagged content like photos, news stories or blog posts. On a typical day online, 7% of internet users say they tag or categorize online content [1]. 20% of adult American users said they maintain a social networking profile, while 8% of users said they maintain a blog [2]. The proliferation of Web 2.0 applications' use in recent years is creating a vast collection of textual entities that interrelate in many ways and forms.

Web 2.0 entities relate to each other in several ways. For example, documents may relate to other documents by referencing each other; a user may relate to a document through authorship relation, as a tagger, as an author, or as mentioned in the page's content. A user may relate to other users through social relations, such as friendship or work. A tag relates to the bookmark it is associated with, and also to the user who added it.

In this paper we present and evaluate novel methods for leveraging social information to enhance search results as well as to discover relations between Web 2.0 entities. The approach we take leverages a unified representation of the entities and their relations. We then use this intricate heterogeneous collection to establish an all encompassing social search solution.

This social search solution allows querying for specific entities as well as retrieving results of all relevant types.In particular, in addition to standard search results, the system returns users related to the query as well as tags that are associated with relevant documents. These tags can be further used to categorize the search results and to better refine the searcher's information need. The system additionally facilitates further exploring the data by enabling queries on the relation between the relevant people and additional content they have contributed. We call such a multi-entity search system based on "social" data a *social search engine.*

The concept of *social search* is used in the literature to describe many different aspects of Web 2.0 search. For example, [13] uses the term *social search* to search over a social network. Such a search is focused on finding a path between users in the network, or finding the set of closest users to a given user in the graph. Commercial people search solutions such as Wink [22], or Spock [19], provide search over social networks as well. The relevant people to any given person are derived directly from users' input, such as social bookmarking and voting.

Many shared bookmarking systems, such as *del.icio.us* [6] provide social search capabilities over the system bookmarks,

[1]Pew Internet & American Life Project. 28% of online Americans have used the internet to tag content. Lee Rainie, January, 2007.

[2]Pew Internet & American Life Project. Digital footprints: Online identity management and search in the age of transparency. Mary Madden, Susannah Fox, Aaron Smith, and Jessica Vitak, December 2007.

users and tags. Users can search over their own bookmarks as well as over the public bookmarks of others. Such systems often provide relevant tags, as well as relevant results to a query. Some systems also provide advanced search capabilities such as searching for similar-minded users based on similarity of bookmarks. However, as far as we know, there is no social search system that provides a unified approach for searching and retrieving entities of all types, as our system does.

1.1 Our approach

In this research we explore novel ways to improve and evaluate the usefulness of *social search* using available social information. The scope of our system and experimentation is in the context of an enterprise, yet the methods we present are generic. The social data we experiment with includes records of users' public activity with documents such as bookmarking, tagging, rating, or comments made to other public Web 2.0 entities. The system we describe allows searching for any object type (e.g document, person or tag) as well as retrieving all entity types for a given query. To this end, the system supports standard textual queries describing a specific information need, entity queries such as searching for a specific user, or any combination of the two. The system retrieves and ranks all the entities that are related to the specified query. The research described in this paper shows that as we hypothesized Web 2.0 information is invaluable in an enterprise environment, where the searchable content is normally of low quality. The large scale experiment described herein is to the best of our knowledge the first such proof point.

In terms of implementation, the *social-search engine* is based on the unified search approach [23, 21, 20]. Unified search, also known as heterogeneous interrelated entity search, is an emerging paradigm within information retrieval (IR). In this approach, the search space is expanded to represent heterogeneous information about objects that may relate to each other in several ways. In addition to the *direct* relations, the method finds *indirect* relations that are implied from the direct relations. For example, documents are directly related to people, and from this relation an indirect relation between people is implied (e.g. a user is indirectly related to another user if both are related to the same set of documents).

Basing the enterprise social search solution on the unified search approach requires a novel embodiment given the demands from such a system. Specifically the system must be scalable, responsive, and reflect the rapid update patterns typical in Web 2.0 systems. Users tag documents and comment on blogs much more often than a Web page is updated. Furthermore, since these same users (and their peers) are the ones that later search for content, they expect to see the social search engine to make use of the new information almost immediately. We describe in this paper a novel realization of the unified search paradigm based on *multifaceted search* [14]. This solution provides an efficient mechanism for updating relations between objects, as well as efficient search over the heterogeneous data. More specifically, our solution represents each of the system's entities (e.g. user, web-page, tag) by a retrievable document, whereas direct relations between entities are represented by marking one of the elements as a "facet" of its counterpart. The strength of the relationship between the two objects is expressed by a new construct that represents the strength of document-facet relationship.

The multifaceted based approach provides an elegant and efficient update mechanism since only direct relations between objects need to be updated when new entities are added (in contrast to methods that model and store all relationships). Indirect relations that are dynamically induced from the direct relations are computed on-the-fly during query execution time. The system is efficient at run time since directly-related objects are retrieved and scored using the search engine's regular scoring mechanisms, while indirectly-related entities are retrieved and scored using an extremely efficient implementation of faceted search (which is required in other domains to quickly count all instances of results that fall into a given facet). In addition to being very efficient, this mechanism allows a truly uniform approach to all entity types and can be easily extended to additional entities.This novel implementation is new and critical to making unified search a viable approach for large scale systems.

To test the validity of the system in the enterprise, we have implemented and deployed the system in our organization. The system has been used by thousands of users on a daily basis, oftentimes replacing the conventional enterprise search engine deployed in the enterprise. We report here on a large user study in which we evaluate the effectiveness of the search results compared to this base enterprise search engine. In addition we evaluate the usefulness and the quality of the related entities that are returned by the social search engine, a feature that is not currently available in traditional search engines. The results are extremely encouraging. The basic search results returned by the social search engine are significantly better than those of the enterprise search engine. Furthermore the related people and tags are shown to be very relevant and accurate to the users query.

To summarize the main contributions of the paper are as follows: (a) Definition of system and algorithms that leverage Web 2.0 information to improve search results in an enterprise setting, and can also be used more generically; (b) A novel realization of the unified search approach, that can scale and meet the demands of a large enterprise search system; (c) A large scale experiment that shows that the proposed approach significantly improves search results as well as enables finding relevant people and tags.

1.2 Paper outline

The rest of the paper is organized as follows: Section 2 briefly summarizes related work, focusing on existing social search solutions. Section 3 describes the implementation of our social search solution using multifaceted search. In Section 4 we describe the implementation of our social search solution over enterprise Web 2.0 data gathered from many sources and applications. Section 5 presents the results of the user study we conducted to evaluate our search engine. Our results demonstrate the effectiveness of social search in the enterprise. We also report on the results of several ranking alternatives for social data. Section 6 concludes and discusses further research directions.

2. RELATED WORK

2.1 Social Search

The advent of Web 2.0 services has opened several new research directions in the IR field related to *social search*. One direction focuses on optimizing Web search using social data gathered from social bookmarking systems [2, 24]. The lesson from these studies is that the set of annotations provided by the public is usually a good summary of the page's topics hence it can be used to enrich the page content. This approach is similar to using anchor text, or other meta-data, as an additional indexing resource for the documents. Heymann et al. [9] analyzed a large sample of bookmarks from the *del.icio.us* social bookmarking system [6] and report that tag terms occur in over 50% of the pages that they annotate, and in only 20% of cases they do not occur in the page text, backlink page text, or forward link page text of the pages they annotate. In addition, the number of annotations of a Web page usually reflects its popularity hence it can be used as an additional evidence of document quality for better ranking of search results.

Social data also enables new search services such as searching for people in a social network [13]. People can search for other people with whom they maintain relationships in the network. The social network is modeled as a graph, where the nodes represent individuals, and a weighted edge between nodes indicates direct relationship between the individuals. In this context, search ranking may depend on the distances among users in the graph.

2.2 Social Ranking

Social ranking, in the context of this work, deals with ranking all entities retrieved by the social search engine. Hotho et al. [10] describe *FolkRank*, a variant of the seminal PageRank algorithm, applied over the multi-entity graph that is associated with the social data extracted from a collaborative bookmarking system. The FolkRank score of an entity relates to the strength of its relations with other entities which are mutually reinforcing each other by spreading activation. Thus, a document tagged with important tags by important users becomes important itself. The same holds, symmetrically, for tags and users.

Similarly, Bao et al. [2] describe another PageRank like algorithm, *SocialPageRank*, which captures the popularity of web pages, users, and annotations simultaneously based on mutual relations. The main assumption behind this algorithm is that "popular" web pages are bookmarked by "up-to-date" users and annotated by "hot" annotations. In addition, this work shows that combining the textual similarity of the tags associated with a Web page to the query, and the *SocialPageRank* score of that page, improves the quality of the search results significantly.

In practice, however, applying PageRank-like computation depends heavily on graph size and is expected to be very slow. Chakrabarti [4] presents *HubRank* which computes and indexes certain random walk fingerprints for a small fraction of nodes in the multi-entity graph. At query time, a small "active" subgraph is identified, bordered by nodes with existing indexed fingerprints. These fingerprints are adaptively loaded and the remaining active nodes are now computed iteratively.

Random walk approaches rank all entities in a uniform manner. However, different entity types provide different retrieval values for the searcher hence they should be ranked according to their own characteristics. Users for example should be ranked according to their authority and to their social activity related to the desired topic, or alternatively, according to their relationship strength with the searcher. In contrast, tags are mostly useful for further refinement of the information need hence they should be ranked accordingly. Web-pages should be ranked according to their relevance to the query, in addition to their authority. In this work we experiment with several alternative ranking approaches, tailored for the different entity types in the system.

2.3 Multi-entity search

Recently, several studies proposed to extend basic search functionality by answering user queries with many types of entities, in addition to the regular relevant Web pages. Multi-entity retrieval is usually based on analysis of the relationships between entities and documents relevant to the query. The work described in [3] studies "object finder" queries, i.e. retrieving the top-k objects mentioned in the set of documents relevant to the query. The score of a target object is determined by aggregating the scores of all relevant documents related to that object. Similarly, the work described in [25] retrieves and ranks named-entities (people, locations), mentioned in the relevant documents to the query. Lou et al. [15] study "relationship queries" which find relations between two entities, e.g., the connections between different places or the commonalities of people. The relationship strength is based on the strength of relations between the Web pages mentioning the two entities. Our approach is strongly related to this direction of research. We also retrieve and rank all desired entities related to the relevant documents. However, our work extends this approach to support searching for all object types and retrieving related objects of all types. Moreover, we will show that our novel implementation which is based on multifaceted search better deals with the efficiency challenges of these new retrieval tasks.

Searching over a *multi-entity graph* generalizes the social search scenarios described above. In such a graph nodes are entities (terms, documents, persons, annotations) and edges are the relations between the entities. Minkov et al. [17] calculate indirect relations between objects by a *lazy random walk* on the graph. The walk propagates relationship strength between nodes – accumulating evidence of relationship over multiple connecting paths. *SimFusion* [23] uses a *Unified Relationship Matrix* (*URM*) to represent this multi-entity graph. Relations between two object types are represented via a relationship matrix M_{ij}. The (k, l) entry of matrix M_{ij} represents the strength of the relation between the object pairs (o_k, o_l) of types O_i and O_j respectively. The *URM* matrix U encapsulates all matrices to provide a unified representation of the unified search space:

$$U = \begin{pmatrix} M_{11} & M_{12} & \dots & M_{1n} \\ M_{21} & M_{22} & \dots & M_{2n} \\ \vdots & & & \\ M_{n1} & M_{n2} & \dots & M_{nn} \end{pmatrix}$$

By iteratively computing over the *URM*, SimFusion integrates relationships from heterogeneous sources when measuring the similarity of any two data objects. Recently, Wang et al. [21] conducted latent semantic analysis over the *URM* matrix, identifying the most salient concepts in the unified semantic space. In a recent work the same authors describe the *SHINE* system which uses an *extended vector space model* for measuring object similarity between users' queries and the information objects, treating both the query and objects as vectors in the unified space [20].

The *URM* matrix provides relationship strength between any two directly related entities, along with a theoretically elegant way to calculate indirect relations through matrix multiplication. However, when implementing a social search system with *URM*, there are two delicate issue to address. The first is that for large multi-entity graphs, computation of indirect relations through matrix multiplication could be computationally expensive. The second issue is the dynamic nature of the system – the rate of creation and update of social relations between objects is expected to be very high, hence efficient dynamic updates of relations between objects should be designed and supported accordingly. In the following we describe the implementation details of our social search solution and how it deals with those two issues.

3. IMPLEMENTATION

Our solution to unified search represents each object in the system in two ways: (1) as a retrievable document, and (2) as a facet (category) of all the objects it relates to. Each direct relation between two objects is thus defined by attaching a facet representing one object to a document representing the other object. The relationship strength between objects is represented by weighting the facet-document relationship.

For example, consider a unified representation of a collaborative bookmarking system. In this system we deal with three object types – bookmarked Web objects (Web-pages), taggers (users), and tags. Each object type is associated with a corresponding document - a Web-page document, a user document and a tag document. The content of a Web-page document would include the content of the Web object it represents, as well as all the tags and the descriptions that users have associated with the Web object. The content of a user document would include publicly available information about the user such as name, title, hobbies, projects, papers. A tag document will contain only the tag. We can consider three obvious relationship types in such a system. Each relation will be represented as a facet as follows:

- The relationship between a user and the tagged Web-page is represented as a *user*-type facet of the corresponding Web-page document.
- The relationship between a tag and the associated Web-page is represented by a *tag*-type facet of the Web-page document.
- The relationship between a user and a tag used for bookmarking is represented as a *user*-type facet of the corresponding tag document.

Recall that the system allows searching for specific objects as well as for all objects relevant to a textual query. When searching for a certain object, the result set will contain all entities related directly as well as indirectly to that object. The directly related objects are extracted by retrieving all entities for which the desired object serves as their facet. Their score is determined according to the strength of the relationship with the target object. When executing a textual query, all directly related objects will be retrieved and scored by the underlying search system, since each object is represented by a textual document. In both cases, the indirectly related objects are extracted by computing the set of facets related to the direct results. In the following we describe the scoring mechanism for indirect objects based on our multifaceted search implementation and show that our scoring mechanism is equivalent to the unified search implementation that is based on matrix multiplication.

3.1 Scoring indirectly related objects

In the unified search framework, the strength of the indirect relation between object o_1 and o_2 is determined by:

$$Score(o_1, o_2) = \sum_{o} U(o_1, o) \cdot U(o, o_2) \tag{1}$$

where $U(o, o')$ is the corresponding entry in the *URM* matrix. This computation is equivalent to squaring the *URM* matrix which provides the relationship strength of order two between any two objects in the multi-entity graph.

Equation 1 can be generalized to score objects based on their indirect relations with any query. For a textual query q, we can score all objects according to their textual similarity to q, since every object is represented by a textual document. For an entity query o we can score all objects according to their direct relationship strength with o. The following score vector, $\vec{s_0}(q)$, provides the (direct) scores of all N objects in the system to the query:

$$\vec{s_0}(q) = (s_0(q, o_1), \ldots, s_0(q, o_N))$$

By multiplying this score vector with U, all objects are scored based on their indirect relationship with q:

$$\vec{s_1}(q) = U \cdot \vec{s_0}(q) \tag{2}$$

Note that Equation 2 can be employed iteratively to compute higher order relationships in the entity graph.

In addition, objects can be scored according to their relative popularity, or authority. Such query independent scores can be determined by the *FolkRank* [10], or *SocialPageRank* [2] algorithms, as described in Section 2, or alternatively by the *inverse entity frequency score (ief)* [25]:

$$ief = \log(\frac{N}{N_o}) \tag{3}$$

where N stands for the number of all objects in the system and N_o stands for the number of objects directly related to o. Similarly to the vector-space *idf* score for terms, the *ief* score "punishes" objects that are related to many objects in general, hence are less specific for a given query. The final score of object o for query q is determined by multiplying the query dependent score with the object static score:

$$Score(q, o) = s_1(q, o) \cdot ief(o) \tag{4}$$

The implementation of Equation 2 requires two stages: 1) retrieve and score objects that are directly related to the query; and 2) multiply the score vector by the *URM* matrix to retrieve indirectly related objects. In order to enable search in reasonable time, this computation must be efficient and scalable. The rest of this section describes how we extend and utilize an efficient multifaceted search implementation in order to retrieve and score indirect results, while still maintaining the sub-second response time that users are expecting from a modern search engine.

3.2 Multifaceted Search

Multifaceted search aims to combine the two main search approaches in IR:

- Navigational Search, which uses a hierarchy structure (taxonomy) to enable users to browse the information space by iteratively narrowing the scope of their quest in a predetermined order

- Direct Search, which allows users to simply write their queries as a bag of words in a text box

In a typical multifaceted search interface, users start by entering a query into a search box. The system uses this query to perform a full-text search, and then offers navigational refinement on the results of that search by categorizing the search results into predefined facets along with the counts of results per facet. Users are able to refine their query by narrowing the search into several of the identified facets. Multifaceted search has become the prevailing user interaction mechanism in e-commerce sites and is being extended to deal with semi-structured data, continuous dimensions, and folksonomies.

Our unified search implementation is based on a *multifaceted search* library [14] developed on top of the open-source Java search engine, Lucene[1]. This library includes several novel features including flexible and dynamic aggregation over faceted data. It not only counts the number of results across several facets, but also supports richer aggregations of numeric and Boolean expressions over the set of results belonging to a given facet. The facets taxonomy is built on the fly, implicitly inferring the facet hierarchy that is inferred by the matching documents.

Rich aggregations and a dynamic taxonomy were prerequisites and a key to implementing the Social Search engine efficiently utilizing a multifaceted search library. Several features however were still missing from the library which were added as part of this implementation:

1. While previously the relation between a facet and a document was binary (the facet could be either associated to the document or not), it was crucial for the system to additionally associate a weight with the relation. This weight is now stored along with the facet.

2. While previously the system returned a non-ranked set of documents per each facet, our implementation requires that the related objects are returned and displayed in ranked order. The system now returns a ranked list of all related documents to a facet query.

3. Given the dynamic nature of Web 2.0 activity, it was important that updates be reflected almost immediately in the index. To support this requirement, we implemented a mechanism that allows facets and numeric fields to be attached to documents that have already been indexed, whereas before they had to be provided at indexing time (which required re-indexing after updates).

4. In order to rank facets by both a dynamic and a static score (e.g. based on popularity) we added a method to associate query-independent static scores with facets. This score is used to associate system objects with their static score.

Our system supports three types of queries: entity queries, textual queries, and hybrid queries. Entity queries consist of a specific object (e.g. a user of the system), the input of textual query is a regular term-based query, using the search engine's query syntax, and the input of hybrid query is a combination of entity and textual queries. All queries return list of directly related documents with associated scores, which are later used to calculate indirect relations. For an entity query, the score reflects the relationship weight between the matching document and the facet representing the query object. For a textual query, the search engine's scoring mechanism is used, and for a hybrid query the results of the sub-queries are combined using the search engine's support of Boolean operators, including separate boosting of sub-queries. Note that for all query types the scores of directly related objects are equivalent to the scores as represented by $\vec{s_0}(q)$.

The score of an indirectly related object, o, is computed by aggregating its relationship strength with all matching documents. multiplied by their direct score:

$$s_1(q,o) = \sum_{i=1}^{N} s_0(q,o_i) \cdot w(o,o_i)$$

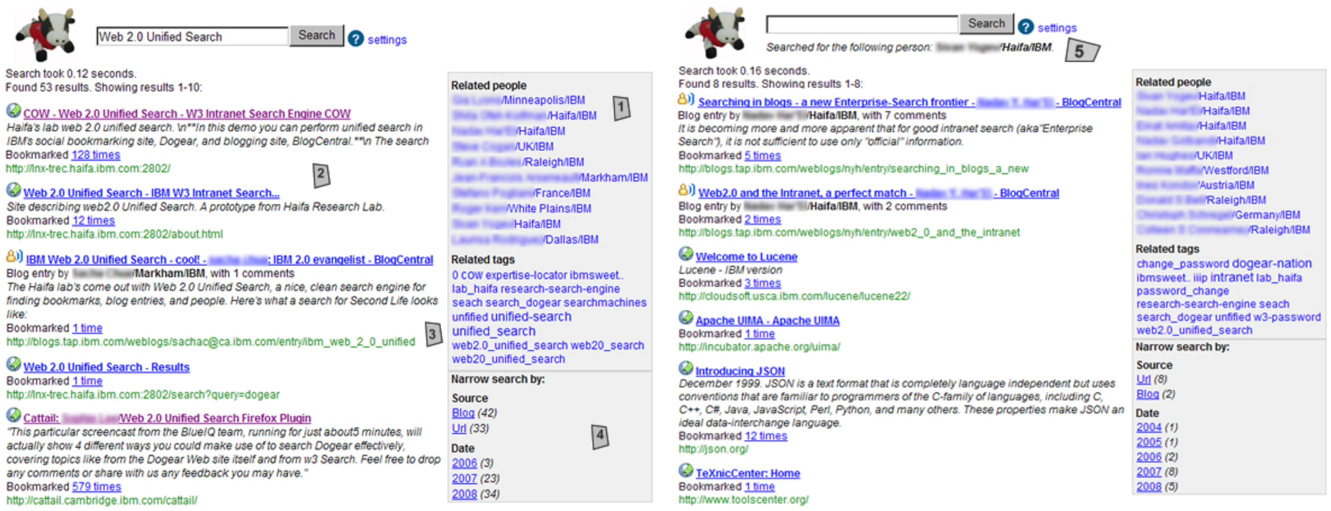
where $w(o,o_i)$ is the relationship strength between the document o_i and its facet o. This computation is implemented efficiently using our faceted-search library's mechanism for aggregating numeric expression over the set of matching documents related to a specific facet. It is easy to see that since $w(o,o_i) = U(o,o_i)$, this facet based computation is equivalent to Equation 2. Finally, the "textual" score of each indirectly related object is multiplied by its static score to determine its final score, $s_1(q,o_i) \cdot \mathit{ief}(o_i)$.

Indirectly related objects are represented by accumulating all facets of the same type. The "user" facet, for example, will include all users that are related to the matching documents, each associated with a score as computed by the facet-based expression. Similarly, each facet, representing one of the other object types, will include all related objects of this type associated with a score expressing their indirectly related strength with the query object.

Note that an object may be related both directly and indirectly to the query object. In this case it is possible to combine the direct score with the indirect score of that object, or maintain separate result sets for directly and indirectly related objects. We implemented the latter, thus the search results include all objects for which their associated documents are directly related to the query, as well as all indirectly related objects which are related to those documents, scored according to Equation 4, and clustered by their entity types.

Figure 1 shows examples of page results of our social search application for a textual query and for an entity query. The directly related documents are given on the left, while indirectly related users and tags are given on the right. By clicking on one of the retrieved users, or tags, the system will execute a new hybrid query narrowing the original results to those that are directly related to the desired object. We describe this application in more details in Section 4 .

Our current application calculates two levels of relatedness. This is however an implementation choice for usability purposes. The mechanism and algorithm enables continuing this computation to as many levels as required by simply using the ranked list of indirectly related objects as the basis of another multifaceted search query, thus traversing paths of length 3 in the entity graph, and this iteration can continue further.



Web 2.0 Unified Search | Search | settings

Search took 0.12 seconds.
Found 53 results. Showing results 1-10:

COW - Web 2.0 Unified Search - W3 Intranet Search Engine COW
*Haifa's lab web 2.0 unified search. \n**In this demo you can perform unified search in IBM's social bookmarking site, Dogear, and blogging site, BlogCentral.**\n The search*
Bookmarked 128 times
http://lnx-trec.haifa.ibm.com:2802/

Web 2.0 Unified Search - IBM W3 Intranet Search...
Site describing web2.0 Unified Search. A prototype from Haifa Research Lab.
Bookmarked 12 times
http://lnx-trec.haifa.ibm.com:2802/about.html

IBM Web 2.0 Unified Search - cool! - [redacted]: IBM 2.0 evangelist - BlogCentral
Blog entry by [redacted]/Markham/IBM, with 1 comments
The Haifa lab's come out with Web 2.0 Unified Search, a nice, clean search engine for finding bookmarks, blog entries, and people. Here's what a search for Second Life looks like:
Bookmarked 1 time
http://blogs.tap.ibm.com/weblogs/sachac@ca.ibm.com/entry/ibm_web_2_0_unified

Web 2.0 Unified Search - Results
Bookmarked 1 time
http://lnx-trec.haifa.ibm.com:2802/search?query=dogear

Cattail: [redacted]/Web 2.0 Unified Search Firefox Plugin
"This particular screencast from the BlueIQ team, running for just about5 minutes, will actually show 4 different ways you could make use of to search Dogear effectively, covering topics like from the Dogear Web site itself and from w3 Search. Feel free to drop any comments or share with us any feedback you may have."
Bookmarked 579 times
http://cattail.cambridge.ibm.com/cattail/

Related people
[redacted]/Minneapolis/IBM
[redacted]/Haifa/IBM
[redacted]/Haifa/IBM
[redacted]/UK/IBM
[redacted]/Raleigh/IBM
[redacted]/Markham/IBM
[redacted]/France/IBM
[redacted]/White Plains/IBM
[redacted]/Haifa/IBM
[redacted]/Dallas/IBM

Related tags
0 cow expertise-locator ibmsweet.. lab_haifa research-search-engine seach search_dogear searchmachines unfified unified-search unified_search web2.0_unified_search web20_search web20_unified_search

Narrow search by:

Source
Blog *(42)*
Url *(33)*

Date
2006 *(3)*
2007 *(23)*
2008 *(34)*

| Search | settings

Searched for the following person: [redacted]*/Haifa/IBM.*

Search took 0.16 seconds.
Found 8 results. Showing results 1-8:

Searching in blogs - a new Enterprise-Search frontier - [redacted] - BlogCentral
Blog entry by [redacted]/Haifa/IBM, with 7 comments
It is becoming more and more apparent that for good intranet search (aka"Enterprise Search"), it is not sufficient to use only "official" information.
Bookmarked 5 times
http://blogs.tap.ibm.com/weblogs/nyh/entry/searching_in_blogs_a_new

Web2.0 and the Intranet, a perfect match - [redacted] - BlogCentral
Blog entry by [redacted]/Haifa/IBM, with 2 comments
Bookmarked 2 times
http://blogs.tap.ibm.com/weblogs/nyh/entry/web2_0_and_the_intranet

Welcome to Lucene
Lucene - IBM version
Bookmarked 3 times
http://cloudsoft.usca.ibm.com/lucene/lucene22/

Apache UIMA - Apache UIMA
Bookmarked 1 time
http://incubator.apache.org/uima/

Introducing JSON
December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.
Bookmarked 12 times
http://json.org/

TeXnicCenter: Home
Bookmarked 1 time
http://www.toolscenter.org/

Related people
[redacted]Haifa/IBM
[redacted]/Haifa/IBM
[redacted]/Haifa/IBM
[redacted]/Haifa/IBM
[redacted]/UK/IBM
[redacted]/Westford/IBM
[redacted]/Austria/IBM
[redacted]/Raleigh/IBM
[redacted]/Germany/IBM
[redacted]/Raleigh/IBM

Related tags
change_password dogear-nation ibmsweet.. iiip intranet lab_haifa password_change research-search-engine seach search_dogear unfified w3-password web2.0_unified_search

Narrow search by:

Source
Url *(8)*
Blog *(2)*

Date
2004 *(1)*
2005 *(1)*
2006 *(2)*
2007 *(8)*
2008 *(5)*

Figure 1: Screen shots of the social search application. Directly related objects on the left and indirectly related objects (users and tags) on the right. (Left): search results for a textual query. (Right): search results for an entity query – all objects related to a specific user.

3.3 Efficiency Factors

Using multifaceted search to model the unified search computation addresses the two issues raised earlier regarding usage of the *URM* matrix for social search: 1) the need for efficient computation of indirect relations and 2) efficient dynamic updates. The efficient runtime calculation of numeric expressions over facets, which in our system is used for the computation of indirect relations, is described in full details in [14]. In general, query execution time is a factor of the extra time it takes to compute the indirect relations. This time increases linearly with the number of the directly related objects to the query. For the social search engine that we have implemented inside our company (described in the next section) that searches over more than 700,000 objects, most queries from all query types (including hybrid queries) are executed in sub-second time on a standard machine (2.8GHz CPU, 2GB RAM). The universal query ($q =' *'$) that essentially retrieves all the objects indexed by the system as well as all objects related to them, query runtime is less than 4 seconds.

Efficient dynamic updates are handled by a mechanism that enables post-indexing addition (and removal) of facets and numeric fields to documents. This mechanism is implemented by storing the changes in an external database. When iterating over facets or numeric fields, database queries are executed in addition to the regular Lucene iteration, and the results of both are combined to supply the most up-to-date data. This data is periodically rolled into the Lucene index in order to utilize Lucene' data structures that are optimized for fast iteration operations. Otherwise, over time we would risk deteriorating query execution due to sub-optimal performance of iterations in the database on large amounts of data. This approach fits very nicely with Lucene's indexing paradigm in which small indexes are created to support rapid updates, and are then merged periodically into the large index to assure optimal query performance. In a similar fashion, we merge the database content into the Lucene index when Lucene performs its index merges, and thus the database that contains only the most up-to-date updates can be kept relatively small.

4. SOCIAL SEARCH WITHIN THE ENTERPRISE

Searching for information within the enterprise is a major concern since inefficient search results in costly productivity loss. Furthermore, effective *people search* services within the enterprise, enables employees to easily identify individuals that are able to assist them with their current tasks. Albeit the advances in search technology over the years, research shows that employees still spend a large percentage of their time searching for information [8]. One of the reasons for the relatively poor performance of existing search solutions in the enterprise is the high complexity of the information space (diverse databases, knowledge management tools, email systems etc.). Moreover, algorithms based on link analysis are much less effective for the Intranet [7].

With the advent and pervasiveness of global enterprises, the need for employees located in sites all over the globe to share information and collaborate is ever increasing. Consequently, Web 2.0 applications are taking off and becoming popular in enterprise settings. These include virtual social communities, wikis, collaborative bookmarking services and blogs. The popularity of these applications outside the firewall additionally contributes to their adoption by employees. Using such applications results in a better sense of community and collaboration. Several Web 2.0 solutions have indeed been developed recently for the enterprise domain (e.g. [16, 11]). In the following we describe the social search solution that we have implemented and deployed. This system leverages the information gathered from existing Web 2.0 applications in IBM to improve enterprise search.

4.1 Social Data

We chose to base our solution on the most used Web 2.0 services that exist within IBM: *Dogear* [16] which is a collaborative bookmarking service used to bookmark and tag pages both within and outside the Intranet; and *BlogCen-*

tral [11] which is a central blog service allowing all IBM employees to publish and manage blogs within the Intranet. We additionally used the enterprise directory and employee profile application, called *BluePages*, to collect information about 15,779 IBMers who are active in these applications. At the time of writing (September 2008), these employees have associated 373,821 bookmarks to 234,856 Web-pages, and authored 77,930 blog threads. Our data thus comprises about 700,000 unique entities.

As described in the previous section, each entity is represented by a retrievable document. Our application uses only the data available through the chosen enterprise Web 2.0 services, and ignores all other sources of information in the enterprise. The content of a bookmarked Web-page includes its content and the users' descriptions and tags as provided by *Dogear*. The content of a blog document contains the post entry, related comments, and the tags associated with the blog. The content of a user document contains the user's public information (name, title, group, related projects), as provided by *Bluepages*, and the content of a tag document contains the tag string itself. Entities are connected by corresponding relation types. Users are connected to the pages they bookmarked, to the tags that they provided, and to their blog entries (as an author or as a commenter). Tags are connected to their related documents. Each relation is stored as a document-facet pair, with relationship strengths expressed by the document-facet weighting mechanism.

4.2 The social search application

The social search application, codenamed *Cow Search*, is available to all users of IBM's Intranet. Figure 1 shows screen shots of the search results for a textual query (left) and for a user entity query (right). On the left of the screen we see the documents that are directly related to to the query (marked by (2)). The "related users" list, marked by (1), shows the list of people found to be related to the set directly related documents. The tag cloud marked by (3) shows the frequency of tags used to describe the set of retrieved documents; and (4) includes additional facets that can be used to further explore the results.

The directly related search results include a mix of blogs, bookmarked pages, and personal profiles (not shown in the figure for privacy reasons). As described in Section 3, for a textual query all relevant documents are scored according to their relevance to the query (as provided by Lucene)[3]. For an entity query (a user or a tag query) documents are scored according to their relationship strength with the queried entity. For all query types the document's textual score is multiplied by the document's independent static score, $ss(d) = \log(X + 2)$, where X is the number of the page's bookmarks and comments.

The list of related users shows people that bookmarked a relevant document, posted a relevant blog entry, or commented on such an entry. Users are scored by Equation 4, while their static score is set according to their *ief* score (see Equation 3). It is important to note that those users are not necessarily experts on the topic. True experts who never bookmarked nor blogged will not be retrieved by our system. However, the set of retrieved users, who found the topic's related material interesting enough to tag it or blog on the topic, can be considered a virtual community of employees sharing common interests on the topic at hand.

[3]Directly related tags are excluded from the list of direct search results.

The set of related tags are represented by a *tag cloud* which is a list of tags related to the retrieved documents. Tags are scored exactly the same as users, according to their relationship strength with the retrieved documents which depends on the number of times the tag has been associated with the document by different users, and according to their *ief* score. The tag score controls its font size in the cloud. In addition the system provides several other facets (marked by (4) in Figure 1) for supporting easy navigation within the search results.

5. USER STUDY

A significant part of this research includes testing the effectiveness of social search in the enterprise. Our goal was to measure both the quality of the returned document set (Web pages, blog entries) as well as the related users and tags. While evaluation methodologies for documents are well known and have standard measures, there are still no standard ways of measuring the quality of related users and tags.

A user study was thus used to measure both aspects of our system. We logged all interactions with the application to obtain information about its usefulness and to trace usage patterns. From the query log of the application we arbitrarily chose 50 queries and ran them to receive a ranked list of 30 documents and the top 100 people for each query, using our baseline algorithm. The documents retrieved were examined and marked with three relevance levels (0-not relevant, 1-marginally relevant, 2-highly relevant) , and the quality of search results was measured by the *normalized discount cumulative gain (NDCG)* measure over the 50 selected queries. NDCG considers the ranking of retrieved objects in addition to their associated relevance levels [12].

To evaluate the effectiveness of the related people results, we emailed the people retrieved by the system a list of the queries they were associated with, and some topics selected randomly, and asked them to rate on a Likert scale of 1 to 5 how relevant they think the topic is to them. We intentionally left the definition of relevance vague to address all kinds of relevance. According to responses we received during the experiment people conceived relevant to them as being relevant to their work in general, their current project, their personal interests, or the interests of their team. We also did not reveal the nature of the experiment or how the topics were generated. All people polled were sent a list of topics they were not related to in addition to those they appeared related to. Thus all people potentially had both relevant and irrelevant topics to rate.

We chose email rather than using a Web survey because we thought people will be more obliged to answer an email directed to them. Emails also allowed us to disassociate the application itself from the topics and hence increasing the likelihood of truthful answers not dictated by our ranking scheme. We sent over 1400 emails for which 612 unique people replied with ratings. Those people come from 116 IBM locations in 38 countries and we assume most of them have no knowledge of each other or of our application.

From the replies we generated 8835 vote pairs of user and self-rating for 50 topics. We thus created a benchmark against which we evaluated our algorithms [4]. We measured

[4]We ignored retrieved users who did not answer the survey.

the NDCG of the related user list, averaged over the set of topics, using the 1-5 scale of user feedback as different relevance levels. For the NDCG calculation we used gains (0,1,3,6,10), for the 5 scale levels respectively, and the discount function used was $-log(rank+1)$.

Despite our survey's breadth, it is somewhat biased by self-rating. Our original attempt to ask people to rate other people's interest in various topics failed, because most respondents simply did not know enough about each other. Consequently we settled for asking people to rate their own interests. Self-esteem and different interpretations of the instructions inevitably lead to different people attaching different meanings to the 5 levels of the rating scale. Some people tend to over-estimate their interest in every topic, while others tend to under-estimate it. In most cases, this issue can be thought of as rating noise that is canceled out by the large number of respondents. But it can still bias some measurements. In particular, our *ief* feature is specifically designed to downplay people who over-represent their interests — contrary to those people's self-rating.

5.1 Results

5.1.1 Social data contribution to enterprise search

In the first experiment we measured the contribution of social data to the quality of enterprise search. Social data affects the search results in several ways. First, social feedback identifies the most authoritative resources in the enterprise, assuming employees comment, bookmark, and tag only high quality pages. Second, the content of a document indexed by our system includes the comments and descriptions people associated with the original document. Therefore, social content is expected to improve search effectiveness similarly to other meta-data resources [7]. Third, a document's static score depends on the number of bookmarks and comments that a page has in our social data collection, therefore, popular documents will be ranked higher.

We measured the quality of search results by manual assessments of the top search results for the 50 chosen queries [5]. Table 1 shows the NDCG(15) and the precision at top k ($p@k$) results, while considering the 3 relevance levels for NDCG computation, and assuming each positive level as relevant for the $p@k$ computations.

	NDCG(15)	p@1	p@5	p@10
1. Enterprise search	0.48	0.44	0.4	0.38
2. Content-Data (CD)	0.29	0.39	0.30	0.26
3. Social-Data (SD)	0.61	0.60	0.60	0.57
4. $SD+$ static-scores(SS)	**0.70**	**0.76**	**0.70**	**0.67**
5. $CD+SD$	0.49	0.45	0.36	0.31
6. $CD+SD+SS$	0.53	0.65	0.56	0.5

Table 1: The precision of the top directly related results, as measured by p@k and NDCG

The first row in the table shows the results of the existing IBM enterprise search engine over the set of topics. This is the official search engine for the IBM Intranet and is regularly used by IBM employees for their daily tasks. This tool indexes all Intranet pages based on their content and page authority, where authoritativeness is determined by link analysis. Currently, it does not exploit any social data, however it includes a feature called quick links, which enables administrators to manually insert search results for popular queries in much the same way that commercial search engines show advertisements. The precision results of the current enterprise search system are given as a basis for comparison.

[5]For each query, all top documents retrieved by all approaches were pulled together and evaluated by the same assessor.

The second row shows the results of our social search engine when only the content of annotated documents is used for indexing (ignoring tags and descriptions provided by the annotators). The third and fourth rows show the results when the same set of annotated documents were indexed using the social tags and descriptions only while ignoring documents' content, without and with static scores respectively (recall that static scores are query independent scores). The fifth and the sixth rows show the results when indexing both social data and content for the same set of documents without and with static scores, respectively.

The first observation from these results, looking at the second row, is that searching only over the content of the annotated documents reduces precision significantly, even below the baseline as determined by the existing enterprise search engine. Focusing the search on the set of annotated documents while ignoring the annotations themselves seems to fail. However, using the annotations for indexing (row 3) and adding page popularity (row 4), improves the precision to be statistically significant better than all other runs (one-tailed paired t-test, $p = 0.002$).

The second observation, in agreement with previous work [2], is that static scores which reflect document popularity improve the search results when combined with textual scores. This is true when adding the static scores to the social-data index (row 3 and 4) as well as when adding static scores to the social-data and content index (row 5 and 6).

The third observation from these results is that adding the annotated data to the raw content of documents improves the search precision significantly (comparing rows 2 and 5). In this experiment we integrated the raw content and the social annotations by concatenating the two texts. However, it seems that this approach does not work well, given that searching over social data only (row 4) is better than searching over content and annotations together (row 6). This is most likely due to the quality of the content (which is very noisy), and to the sub-optimal weighting of the social data compared to the content. The challenge of finding an optimal integration policy for content and social annotations is left for future work.

The significant improvement in precision when using social information to rank pages compared to the baseline, clearly demonstrates the value of the social search engine. While existing enterprise search solutions struggle with noisy datasets and lack of link data and thus have difficulties in retrieving high quality results, the documents annotated by social data gathered by the Web 2.0 tools enable the social search engine to take advantage of the "wisdom of crowds" and provide much more precise results.

5.1.2 Related users

The second experiment evaluated whether users retrieved by the social search system are indeed related to the submitted queries. We used the results of the user study described above where each retrieved user was asked about his interests in a list of topics containing the topics he was associated with, and a list of random topics he was not associated with. Table 2 shows the NDCG of the top k users, $k = 10, 20, 30$, measuring the agreement of retrieved users with the system judgment (i.e. ranking) of their relatedness to the searched topics. We compare several ranking schemes. The first row provides the results when users are ranked by simply counting the number of the documents related to them in the result set. The second row shows the results when users are ranked by summing the score of documents related to each of the users. The third row shows the results when associating "optimal" weights to the relation types between users and documents. Optimal weights were found by exhaustive search using the *Downhill simplex method* [18], using a sample of the marked data for training. The optimal relative weights found were (1, 3.1, 0) for the relation types ("tagger", "blogger", "commenter"), respectively. In the last row we multiplied the user's score by the *ief* static score. This row represents the full user scoring mechanism as expressed by Equation 4.

	NDCG		
Ranking	10	20	30
1. count-only	0.71	0.69	0.68
2. sum of doc scores	0.75	0.73	0.72
3. +relationship weighting	0.76	0.74	0.73
4. +user's *ief*	**0.77**	**0.76**	**0.74**

Table 2: The agreement between retrieved users and the system ranking of their relatedness to the searched topics, as measured by NDCG of top k results

There are several interesting insights from these results. First, the results exhibit increasing agreement as we consider document scores, optimal relative weights for the different relation types, and user static scores. The results while using the full scoring function, including users' *ief* (row 4), are statistically significantly better than count-only and sum of doc-scores (one-tailed paired t-test, $p = 0.05$), but are not significantly better than ignoring user static scores (row 3). Second, the optimal weight for commenter-document relation type in this experiment was found to be zero. This means that users who comment on a blog entry are not necessarily related to the blog's topic. While we do not have full explanation for this result, we speculate that there are users who comment to a blog not because of its content but rather because of its popularity. This phenomenon should be further investigated.

5.1.3 Related tags

The last experiment measured whether tags retrieved by the social search system are indeed related to the submitted queries. The evaluation was done using the *Normalized Google Distance* (NGD) [5], which is a measure of semantic interrelatedness between terms derived from the number of hits returned by a search engine. Terms with the same or similar meanings in a natural language sense tend to be "close" while words with dissimilar meanings tend to be farther apart.

For each query we found a large number of related tags using several different tag scoring formulas. Then, for each query-tag pair, we performed the following three queries in IBM's enterprise search engine: a) the query alone, b) the tag alone and c) the query and tag together. The number of results for each of the queries along with the total number of documents in the search index defines the NGD between the query and the tag. NGD scores are in the range [0,1], with lower values for closer terms. In order to perform NDCG tests, we translated the (continuous) NGD score to a (discrete) relevance level as follows: each NGD score was multiplied by 10, rounded to the closest integer, and then subtracted from 10. The resulting scores are integers from 0 to 10, with higher values for closer terms. Thus, tags are marked by a relevance degree according to their semantic distance from the query.

Table 3 shows the NDCG of the top k tags, $k = 10, 30, 50$, measuring the agreement between the relevance level of retrieved tags with the system judgment (i.e. ranking) of their relatedness to the searched topics. We compare the results using different ranking schemes. The first row provides the results when tags are ranked by only counting the number of their related documents. In the second row we ranked tags by summing the score of related documents to each of the tags, and in the third the score is multiplied by the tag's *ief*. In the last row, in addition to multiplying by the *ief*, we multiplied each related document score by the number of times the tag is used to describe that document. This row represents the full user scoring mechanism as expressed by Equation 4.

	NDCG		
Ranking	10	30	50
1. count-only	0.617	0.688	0.736
2. sum of doc scores	0.617	0.689	0.736
3. +*ief*	**0.681**	**0.745**	**0.777**
4. +relationship weighting	0.672	0.753	0.784

Table 3: The agreement between the semantic distance of retrieved tags to the query with the system ranking of their relatedness, as measured by NDCG of top k results. Tag semantic distance to the query was measured by NGD.

The results show a very high correlation between the tags and the query, thus corroborating the hypothesis that tags related to the set of documents are also highly related to the query. The results also show that the tag's *ief* is an important factor in improving tag retrieval. There are several tags that are very popular overall, but do not distinguish one query from the other (e.g. "IBM"). The results while using ief (row 3), are statistically significantly better than all other runs (one-tailed paired t-test, $p = 0.05$). The last row shows that the agreement is slightly lower when using the relationship weight between documents and tags.

6. SUMMARY

This work addresses new ways for augmenting search and discovery using multiple types and sources of social information. We described a social search solution using a unified approach in which all system entities are searchable and retrievable. Our solution has been implemented using a multifaceted search library which provides an elegant solution to discovering indirect relations between objects.

Our social search engine has been deployed within a large enterprise, applied over social data gathered from several Web 2.0 applications. We conducted a large user study to evaluate the contribution of social data for search. The results reveal that social data is valuable in several ways. At first, the high precision of top retrieved documents demonstrate that user feedback identifies high quality content in the corpus thus focuses the search to valuable data only. Second, user comments and tags are highly valuable in general and augment the description of system entities, as well as providing additional evidence for object popularity. The results of the user study we conducted confirmed that users related to a query, as determined by the system, indeed have interest with the topic they were retrieved for. Similarly, social tags retrieved by our system are strongly related to the query, as measured by a semantic distance measure.

Our social search framework can be extended in several directions. Emerging object types and relations in new coming social services can be easily integrated into our system. In particular, social networks that provide personal relations between individuals can be further exploited for search personalization. Personal relations can bias the ranking of entities related to the searcher's close community. For example, related users can be ranked according to their distance from the searcher in the social network graph, in addition to their relationship strength with the query. Our system can also be used for recommendation. Documents can be recommended to a searcher based on their relationship strength with similar users with common interest. Similarly, tags can be recommended to a tagger based on their relations with similar taggers, or with similar documents.

Another direction for further research is to quantify the contribution of social objects to the effectiveness of the search system. While related users and tags seems to be very valuable, as complementary facets to regular search results, we still do not have good evaluation methodology to measure their direct contribution. This issue is still open and left for further research.

7. REFERENCES

[1] Apache Lucene. http://lucene.apache.org/java/docs/.

[2] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 501–510, New York, NY, USA, 2007. ACM.

[3] K. Chakrabarti, V. Ganti, J. Han, and D. Xin. Ranking objects based on relationships. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 371–382, New York, NY, USA, 2006. ACM.

[4] S. Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 571–580, New York, NY, USA, 2007. ACM.

[5] R. Cilibrasi and P. M. B. Vitányi. Automatic meaning discovery using google. In *Kolmogorov Complexity and Applications*, 2006.

[6] del.icio.us. http://del.icio.us/about/.

[7] P. A. Dmitriev, N. Eiron, M. Fontoura, and E. Shekita. Using annotations in enterprise search. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 811–817, New York, NY, USA, 2006. ACM.

[8] D. Hawking. Challenges in enterprise search. In *ADC '04: Proceedings of the 15th Australasian database conference*, pages 15–24, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[9] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarks improve web search? In *WSDM '08: Proceedings of the First ACM International Conference on Web Search and Data Mining*, New York, NY, USA, 2008. ACM.

[10] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *ESWC '06: , Proceedings of the 3rd European Semantic Web Conference*, pages 411–426, 2006.

[11] J. Huh, L. Jones, T. Erickson, W. A. Kellogg, R. K. E. Bellamy, and J. C. Thomas. Blogcentral: the role of internal blogs at work. In *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, pages 2447–2452, New York, NY, USA, 2007. ACM.

[12] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions of Information Systems*, 20(4):422–446, 2002.

[13] J. Kleinberg. Social networks, incentives, and search. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 210–211, New York, NY, USA, 2006. ACM.

[14] R. Lempel, O. Ben-Yitzhak, N. Golbandi, N. Har'El, S. Yogev, D. Sheinwald, B. Sznajder, S. Ofek-Koifman, E. Shekita, and A. Neumann. Beyond basic faceted search. In *WSDM '08: Proceedings of the First ACM International Conference on Web Search and Data Mining*, New York, NY, USA, 2008. ACM.

[15] G. Luo, C. Tang, and Y. li Tian. Answering relationship queries on the web. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 561–570, New York, NY, USA, 2007. ACM.

[16] D. R. Millen, J. Feinberg, and B. Kerr. Dogear: Social bookmarking in the enterprise. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 111–120, New York, NY, USA, 2006. ACM.

[17] E. Minkov, W. W. Cohen, and A. Y. Ng. Contextual search and name disambiguation in email using graphs. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 27–34, 2006.

[18] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1994.

[19] Spock. http://www.spock.com.

[20] X. Wang, J.-T. Sun, and Z. Chen. SHINE: Search heterogeneous interrelated entities. In *CIKM'07: Proceedings of the 29th annual international ACM SIGIR conference on Information and Knowledge Management*, New York, NY, USA, 2007. ACM Press.

[21] X. Wang, J.-T. Sun, Z. Chen, and C. Zhai. Latent semantic analysis for multiple-type interrelated data objects. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 236–243, New York, NY, USA, 2006. ACM Press.

[22] Wink. http://wink.com/wink/about.

[23] W. Xi, E. A. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan, and D. Zhuang. SimFusion: measuring similarity using unified relationship matrix. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 130–137, New York, NY, USA, 2005. ACM Press.

[24] Y. Yanbe, A. Jatowt, S. Nakamura, and K. Tanaka. Can social bookmarking enhance search in the web? In *JCDL '07: Proceedings of the 2007 conference on Digital libraries*, pages 107–116, New York, NY, USA, 2007. ACM.

[25] H. Zaragoza, H. Rode, P. Mika, J. Atserias, M. Ciaramita, and G. Attardi. Ranking very many typed entities on wikipedia. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 1015–1018, New York, NY, USA, 2007. ACM.