Finding People and Documents, Using Web 2.0 Data

Einat Amitay
einat@il.ibm.com

David Carmel
carmel@il.ibm.com

Nadav Golbandi
nadav.golbandi@gmail.com

Nadav Har'El
nyh@il.ibm.com

Shila Ofek-Koifman
shila@il.ibm.com

Sivan Yogev
sivany@il.ibm.com

IBM Haifa Research Lab
Haifa 31905, Israel

ABSTRACT

Given a user's free-text query, search engines return a ranked list of documents that are likely to be helpful to the user. In this research, we propose a simple yet highly effective technique for also providing a ranked list of *related people* to every search. The list of people related to the query is calculated at search time using an enhanced *faceted search* engine, based on person-document relationships mined from several Web 2.0 applications (such as blogs and social bookmarks) in the intranet of a large enterprise.

Our hypothesis is that the *related people* we retrieve for a query are people who have special interest in the query's topic, and thus may be useful to the person making this query. We conducted a large user study with over 600 people to confirm this hypothesis.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Social Search, Faceted Search, Enterprise Search

1. INTRODUCTION

When they are in need of information, some people like to find a written document which explains what they want to know. Yet, other people prefer to find the right person — one who might know the answer to their question — and ask him or her for the specific information they need. Most people are somewhere between these extremes, preferring to find documents in some cases, and people to ask in other cases. Also, for some topics, only one of these information sources is available. This is why we believe that search engines should provide both types of results: Given a query, the search engine should provide both a ranked list of documents that might answer the user's query, and a ranked list of people that are interested in the query's topic, and might be able to help.

Future Challenges in Expertise Retrieval
SIGIR 2008 workshop, July 24, Singapore

Often the data and techniques used to find relevant documents, and those used to find relevant people (as in expert search), were separate and unrelated. In this paper, we propose a *unified* method that finds both relevant documents and relevant people for every query.

As we shall see in section 2 below, the key to our technique is knowing for each document which person is *related* to it. An excellent source for both documents and document-person relationships are so-called Web 2.0 applications, such as blogs and social bookmarking systems. In addition to the actual documents, these applications can tell us who is related to each document, and in what way. For example, a person can be related to a blog entry as its *author* or as a *commenter*, and can be related to any page as a *bookmarker*.

We will then show how to use this *social information* — documents and document-person relationships — to determine which people are most relevant to a given query. We will use an enhanced faceted search engine to determine the (potentially large) set of relevant documents for this query, and then which people are most related to these documents.

In this work, we focused our attention on the case of *enterprise search*, i.e., search in the intranet of a large organization. Compared to the open Internet, people in the enterprise are easier to track (because they use the same user-id everywhere), and are more likely to be helpful to each other.

In section 3, we evaluate the validity of the "related people" results. The individual document-person relationships (author, commenter, bookmarker) indicate that the person is in some way *relevant* to the content of the document; So it is natural to hypothesize that the query-person relationship we derive from them also measures the person's relevance to the topic of the query. We checked this hypothesis using a large user study with over 600 participants.

2. THE SYSTEM

2.1 Social Information

Traditionally, building a Web site took a considerable amount of effort and expertise, so most users were relegated to the role of information consumers, not producers. Web 2.0 services, such as forums, wikis, collaborative bookmarking services, and many more, allow ordinary users to become information producers. In turn, this allows people to learn from the experiences and knowledge of their peers — something which is especially important in the intranet of a large international enterprise.

Web 2.0 sources not only provide a new wealth of infor-

mation, they also provide new types of information, which we call *social information*. The new types of information include user-supplied metadata for documents (bookmarks, tags, ratings, comments), relationships between people and documents (who wrote a document, who commented on it, who tagged it, and so on), and other relationships such as between people and people, or between documents and tags.

The goal of a *social search engine* is to use the social information to improve the user's search experience over regular full-text search. One way of improving search is to improve the relevance of document results [3, 16, 6]: Tags (and other forms of comments) supply more text that can be considered during search, and important documents can be recognized by the amount of user activity around them (such as the number of times they were bookmarked or commented on).

But using the social information, we want to do more than just return better documents. The literature [5, 17, 10, 13, 15, 14] proposes the idea of *multi-entity search*, where other entities besides documents can be used in queries or turn up in search results. In our case, we want *people* to be searchable entities in our system, exactly like documents: Related people will be returned for every query (in addition to the relevant documents), and people can also be used as query terms.

2.2 Related People

In the standard vector space model of IR, each document is represented as a normalized vector that measures the relevance of each term (word) to the document. The entire document collection is therefore represented as a relevance matrix D between documents and terms; D_{ij} is the relevance of the i^{th} document to the j^{th} term. A query is represented, just like a document, as a vector q. The product Dq is a vector giving the relevance of each document to the query q. I.e., these are the search results.

The people-document relationships in the social information allow us to define a second relevance matrix P, between documents and people. P_{ij} measures the relevance of the i^{th} document to the j^{th} person. We might, for example, want to give a high relevance P_{ij} when person j wrote document i, a lower relevance if they commented it, and a lower still relevance if they merely bookmarked it.

Multiplying the term-document relevance matrix and the document-person relevance matrix yields a term-person relevance matrix P^TD that can be then be used in search: P^TD can be, just like D, multiplied by a query vector, resulting this time in relevant people (instead of documents). The relevance of these people to the query is *indirect*, through the documents — a person is relevant to a query if he or she are relevant to documents which are relevant to the query.

The need to calculate the matrix product P^TD causes problems, though. Every change to the social information can require modifying large parts of this matrix, making it difficult to index dynamically-changing data. It also means that searching for relevant documents and relevant people is done separately, using two different relevance matrices (D and P^TD). Finally, most search engines offer capabilities beyond the simple vector space model (e.g., supporting searches of multi-word phrases, considering term proximity, and more), and using the matrix P^TD directly forces us to give up on these features.

We therefore propose an alternative technique which (as we shall show) gives equivalent results, but solves all the above problems. The idea, already found in [2, 11], is to first use the given search engine to find the relevant documents; Then, knowing which people are relevant to each of these documents, we start aggregating the relevance of each person. This process can be realized using *faceted search*, with the related people added as a *facet* to each document:

2.3 Faceted Search for Related People

Faceted search is a commonly-used technique for adding navigation to a search engine. A *facet* is a single attribute of the document, e.g., in a book search application there might be an "Author" facet and a "Price" facet, and in our application there is a "Related Person" facet. Faceted search starts, like ordinary search, by finding all documents matching the user's query. But while an ordinary search system will only show the few documents with the highest relevance, a faceted search system goes over all matching documents, counting the number of documents found for each subcategory of the facet (individual authors, price ranges, etc.), and finally displays the categories with the highest counts.

Our unified search solution is based on a faceted search library [4] developed upon the open-source Java search engine, Lucene [1]. This library has several simple but useful extensions to the faceted search paradigm, which we shall use. For the purpose of this work, the two most important extensions are these:

- Instead of just counting the number of documents for each category, the library can aggregate other numeric expressions. E.g., the sum of these documents' relevance score to the query.
- The relation between a category and a document is not just binary (it is either attached to the document, or not) — it can be assigned a weight.

These capabilities are exactly what we need to produce related-people scores which are identical to the scores that the matrix approach described above would have produced: As explained above, given a query vector q, $(P^TD)q$ is a vector specifying for each person, his or her (indirect) relevance to the query. Let's rewrite this multiplication as $P^T(Dq)$. But Dq is nothing more than the vector of matching documents, specifying the relevance score of each document to the query. Looking at position i of this vector equality, we therefore discover that the indirect relevance of person i to the query (according to the matrix method) is identical to

$$\begin{aligned}\Big((P^TD)q\Big)_i &= \Big(P^T(Dq)\Big)_i \\ &= \sum_{j=1}^{\text{ndoc}} (P^T)_{ij}(Dq)_j \\ &= \sum_{j=1}^{\text{ndoc}} P_{ji} \cdot \text{score}_q(\text{document } j)\end{aligned}$$

If we remember that the relevance score is non-zero only for matching documents, and that P_{ji} is the known relation strength between document j and person i, we end up with the formula (as proposed in [2] with different justification):

$$= \sum_{\substack{\text{matching} \\ \text{documents } d}} \text{relation}(d, \text{person } i) \cdot \text{score}_q(d)$$

The extended faceted search indeed allows aggregating this sum for each person i (i.e., each category of the related people facet). relation$(d, \text{person } i)$ is available as the weight of the person-i category on document d, and score$_q(d)$ is available for each document because the facet aggregation starts after the document relevance scores have already been calculated.

The faceted search library of [4] contains two further extensions which are useful for our social search application:

The library allows associating with each category (in our case, person) a query-independent static score (or category *boost*). The final score of each category (person) is determined by multiplying its query dependent score with its static score. The static score of each person can be defined according to their relative popularity or authority, e.g., using the FolkRank score [7]. In our implementation, we chose to use *inverse entity frequency* (*ief*) [17]. It is defined as

$$\text{ief}(person) = \log(\frac{N}{N_{person}})$$

where N stands for the number of all documents in the system and N_{person} stands for the number of documents related to this person. Analogous to the *idf* score for terms, the *ief* score "punishes" categories that are related to many documents in general, hence are less specific to a given query.

The last faceted-search extension of interest is using the category weights to score the documents when searching for all documents in a certain category. In our social search application, this means that the top results for "All documents related to person P" will be documents which the person wrote, rather than merely commented on or bookmarked.

We've already seen that documents and people have an equal standing in our system when it comes to the search results (results of both types are returned for each query). The same is true for queries: in addition to textual queries, we can search by person (as explained in the previous paragraph), or use a combination of text and people as a query.

2.4 The Social Search Application

In the following we describe our social search implementation, based on social information gathered from IBM's intranet. From the internal Web 2.0 services in IBM, we chose the currently most used ones: *Dogear* [12], a collaborative bookmarking service used to bookmark and tag pages both within and outside the intranet; and *BlogCentral* [8], a blog service allowing all IBM employees to manage blogs within the intranet. We also used the enterprise directory application, called *BluePages*, to collect information about the IBMers who participated in Dogear or BlogCentral. At the time of writing, 15,779 employees assigned 337,345 bookmarks to 214,633 Web-pages, and wrote 67,564 blog threads.

The content we indexed for a Web page contained its title and the users' descriptions and tags as provided by *Dogear* (the actual content of the page was *not* crawled and therefore not available). For blogs, the indexed document was a blog thread, containing the blog entry, comments, and tags. For each person, we had a document containing the person's directory information (such as name, title, and group). Finally, people were connected, as facets, to the pages they bookmarked, and to their blog entries (as an author or as a commenter). Tags are connected to their related documents.

A static-score (or *boost*, in Lucene nomenclature) was given to each document based on the amount of activity around

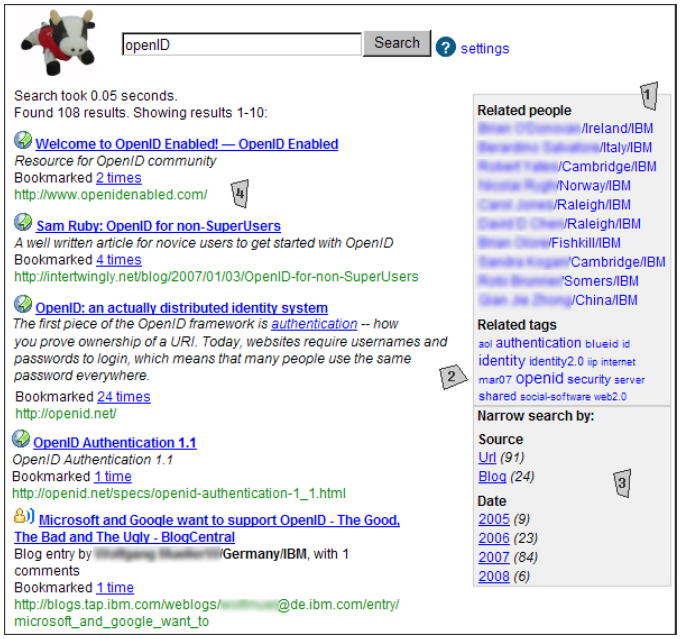

Figure 1: The social search application

it. In essence, a page which was bookmarked by many people, or a blog entry that was heavily commented or rated, is more likely to be a good search result than a document in which hardly anyone expressed interest. The actual boost used was $\log(X + 2)$, where X is the number of bookmarks, ratings and comments on that document. Our evaluation showed that this boosting significantly increased the document search precision.

The social search Web application, codenamed *Cow Search*, was made available to all users of IBM's intranet. Figure 1 shows a screenshot of the application, given the query "openID". On the left (marked by <4>) we see the most relevant documents — a mix of blogs, Web pages and personal profiles (not shown in the figure for privacy reasons). On the right <1> is the list of related people, calculated as described in the previous section. The "Related tags" tagcloud <2> is calculated in a similar manner — each tag is a category, and the weight of the association of a tag with a document is the number of times this tag was used to describe this document. <3> shows some additional facets which aid navigation within the search results.

The list of "Related people" is not necessarily the list of IBM's experts on the topic. Experts who never bookmarked or blogged obviously cannot be retrieved by our system. Rather, the "Related people" are people that expressed interest in the topic — bookmarked a relevant document, posted a relevant blog entry, or commented on such an entry. In the next section (Evaluation) we show using a large survey that people generally agree with the system's determination of how "related" they are to various queries.

As explained in the previous section, a query can be either textual (in which case documents are matched and scored using Lucene's search algorithms), or a reference to a person (in which case the documents related to this person are returned, scored according to the relationship strength). Hybrid queries, containing both text and a reference to one or more person, are also possible.

Person and hybrid queries have a number of interesting uses. For example, giving a person as a query (or using a "everything related to this person" link) yields not only the documents related to this person, but as usual also shows related people, i.e., people who were interested in the same documents as the given person. Another useful example: For any query, if person X is a "Related person", then adding this person X to the query will find the intersection of the documents that matched the original query with the documents related to X, which is essentially the *evidence* of why person X turned up as a related person in the first place. In our application, the user can click on each of the names in the "Related people" list and choose "why this person?", to run this hybrid query. Users find this sort of evidence an important feature of the application. Another link for each person, "who is this?", displays directory information.

3. EVALUATION

Our unified search system returns both documents and people for every query. Therefore, to evaluate the quality of our system we needed to evaluate the quality of both lists.

We evaluated the document results using standard IR evaluation methodology — running 50 example queries, and having the results be judged by humans. We found the document results to be of very high quality (e.g., P@10 was 0.81). The very high precision of the top results demonstrates the capability of the social search engine to focus on good resources from the entire collection, while existing enterprise search solutions struggle with noisy datasets and have difficulties in retrieving high quality results. However, the document results are outside the scope of this workshop and therefore we will not go into details about this evaluation. Rather, in this section, we will describe in detail a large user study that we performed to evaluate the quality of the "related people" list.

From the log of queries submitted by real users to the application, we arbitrarily chose over 60 queries and ran them to receive a ranked list of 100 people for each query, using our baseline algorithm.

We then emailed all these people a list of 6-15 queries, which we defined as topics, to rate on a Likert scale of 1 to 5 whether they think the topic is relevant to them or not. We intentionally left the definition of relevance vague to address all kinds of relevance. According to replies we have received during the study, people conceived relevant to them 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 where the topics were generated from. All of the people received along with the topic they appeared related to, a list of topics they were not found related to, thus all 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 have sent over 1400 emails for which 612 unique people replied with ratings. Those people came 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 60 topics. We thus created a benchmark against which we evaluated our algorithms. To quantify our results' agreement with the benchmark, we used *normalized discounted cumulative gain (NDCG)* [9], which measures a ranked list's agreement with known 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 $-log(rank + 1)$. The NDCG scores we report below are an average over the set of topics.

Despite our survey's breadth, it is to some degree biased by self-rating. Our original attempt to ask people to rate other people's interest in various topics had failed, because most respondents simply did not know enough about each other. This is why we had to ask people to rate their own interests. But 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 — and therefore we expect this survey not to measure the full value of this feature.

3.1 Results

	NDCG		
Ranking	10	20	30
count-only	0.71	0.69	0.68
sum of doc scores	0.75	0.73	0.72
+relationship weighting	0.76	0.74	0.73
+person static-score using *ief*	0.77	0.76	0.74

Table 1: The agreement of retrieved people with the system ranking of their relatedness to the searched topics, as measured by NDCG of top k results

Table 1 shows the NDCG of top k people, $k = 10, 20, 30$, measuring the agreement of retrieved people with the system's judgment of their relatedness to the searched topics. The different rows in the table show the agreements as progressively more and more features of the faceted search system were employed: The first row provides the results when people are ranked by just counting the number of their related documents (this is the "Votes" method of [11]). In the second row we ranked by summing the score of related documents to each of the people ("CombSUM" of [11]). The third row shows the results when associating different weights with the different relation types between people and documents (as in [2]); By exhaustive search we found the optimal weights to be (1, 3.1, 0) for the relation types ("tagger", "blogger", "commenter") respectively. Finally, in the last row we add the *ief* static-score for people. This row represents our full person-scoring mechanism as described in section 2.

There are several interesting insights from these results. First, the results exhibit better agreement as we consider document scores, optimal relative weights for the different relation types, and static scores for people. Second, the optimal weight for the relation between a blog entry and a commenter was found to be negative in this study (above, we used a zero weight instead, which was slightly sub-optimal).

This means that a person who comment on a blog entry is actually (slightly) *less* likely to be interested in the entry's topic than a person who did not comment on that entry. While we do not have full explanation for this result, we speculate that there are people who comment to a blog not because of its specific content but rather because of its popularity. This phenomenon should be further investigated.

4. SUMMARY

In this research, we proposed a simple yet highly effective method for a search engine to return both relevant documents and relevant people for every query. The method also allows queries to contain people instead of, or in addition to, textual terms. The list of people related to the query is calculated at search time using an enhanced faceted search engine, based on person-document relationships mined from Web 2.0 applications.

We proved that our faceted-search based method gives identical results to a more recognizable vector space model, but showed that our method has unique advantages — such as working efficiently with dynamically-changing data and taking advantage of advanced features of existing search engines (e.g., phrase search) that are not part of the classic vector space model.

We described an implementation of our method, a *social search engine* called "Cow Search" deployed in IBM's intranet. The social search engine provides several unique features not found in standard enterprise search solutions: It returns higher quality documents, as well as related people, for every query. It also allows referring to people (not just textual terms) in queries — a feature that has several interesting applications (e.g., it can be used to show the *evidence* that lead us to believe that a certain person is related to a given query).

Informal feedback from users of "Cow Search" has been very positive; Users were very pleased with the quality of both document and related-people results. We also conducted a large-scale formal evaluation. We measured the precision of the returned documents to be exceptionally high (P@10 = 0.81), and conducted a large user study, with over 600 respondents, to measure how much people agree with which topics our algorithm said were related to them. The results of this study show high agreement, and that the full method which we described is clearly better than more naive methods, such as using counting-only faceted search.

5. REFERENCES

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

[2] K. Balog and M. de Rijke. Finding experts and their eetails in e-mail corpora. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 1035–1036, 2006.

[3] 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, 2007.

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

[5] 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, 2006.

[6] 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*, 2008.

[7] 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.

[8] 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 extended abstracts on Human factors in computing systems*, pages 2447–2452, 2007.

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

[10] 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, 2007.

[11] C. Macdonald and I. Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 387–396, 2006.

[12] 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, 2006.

[13] 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.

[14] 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*, 2007.

[15] 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, 2005.

[16] 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, 2007.

[17] 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, 2007.