

FacFinder: Search for Expertise in Academic Institutions*

Yi Fang^a, Luo Si^{a,**}, Aditya Mathur^a

^aDepartment of Computer Science, Purdue University, West Lafayette, IN 47907, USA

Abstract

Interdisciplinary collaboration and external funding opportunities have pushed academic institutions to take steps to advertise their most important organizational resource: faculty expertise. This paper presents FacFinder— a publicly accessible faculty expertise search and ranking system across multiple universities. The key components of FacFinder are exposed and discussed along with the underlying rationale and design decisions. Also presented are the results of experiments aimed at evaluating the effectiveness and efficiency of FacFinder in its operational environment.

Key words: Faculty expertise search, Information retrieval, Expert finding

1. Introduction

Rapid growth of technical knowledge has prompted faculty from various fields to join in addressing complex problems that cut across traditional disciplines. Many of today's scientific grand challenges in fields such as nanotechnology, bioinformatics and neuroscience demand interdisciplinary participation and collaboration, but manually locating the collaborators with required expertise can be difficult and time-consuming, especially for planners such as those in government agencies or universities. In addition, funding agencies and industry sponsors need an easy-to-access system to locate potential faculty to develop joint R&D efforts. Students are also avid seekers for prospective advisers with matched research interests. In response, academic institutions are developing directories and databases to allow users to browse the research interests and expertise of their faculty.

This paper describes the underlying rationale and design of FacFinder, which is a search engine in use on the Indiana Database of University Research Expertise (INDURE)¹. FacFinder is a scalable information retrieval (IR) system for ranking faculty across multiple universities based on their research expertise. To the best of our knowledge, FacFinder is the first search engine of its kind in the public domain. Over 12,000 faculty are currently searchable through the system. INDURE is evolving everyday to include additional information about faculty. To ensure accuracy in ranking faculty with a given expertise, we have designed a set of novel techniques including the following: 1) multiple indexes are built by considering the heterogeneities of the different information sources instead of the single index approach used by most IR systems; 2) adaptive parameters are explored in the system to address different types of queries without human intervention; and 3) original user queries are transformed by taking the term proximity and ordering into account.

*This work was supported in part by a grant from the Indiana Economic Development Corporation.

**Corresponding Author: Luo Si

Email addresses: fangy@cs.purdue.edu (Yi Fang), lsi@cs.purdue.edu (Luo Si), apm@cs.purdue.edu (Aditya Mathur)

¹<http://www2.itap.purdue.edu/indure/>

In the remainder of this paper, we investigate the faculty expertise search problem as well as the various design and development issues and present the solutions we are pursuing. Section 2 provides a brief description of the related work. In Section 3, we highlight some of the technical challenges of expertise search in academic institutions. Section 4 introduces the individual components of FacFinder. Section 5 describes our experimental methodology to evaluate the performance of the system and Section 6 presents the experimental results. In Section 7, we summarize the current work and mention some challenges to be met in the near future.

2. Related work

Expert finding in enterprises started gaining popularity towards the end of the '90s. Several expert locator systems have been developed in business organizations to help employees collaborate. For example, the CONNEX system developed by Hewlett-Packard (Davenport , 1996) was a project to provide a guide to human resource within the company. Microsoft's SpuD (Skills Planning and Development) project (Davenport and Prusak , 1998) was to develop a database containing job profiles across the IT group and to help match system developers' expertise with the jobs. The goal of National Security Agency's KSMS (Knowledge and Skills Management System) (Wright and Spencer , 1999) was to categorize the talent pool within the Agency to allow precise identification of knowledge and skills. More recent notable examples of systems include MITRE's Expert Finder (Maybury , 2006) and IBM's SmallBlue (Lin et al. , 2008). Expert finding systems also attract close attention of the IR research community. The expert search task is a part of the Enterprise track² of the Text REtrieval Conference (TREC) since its first run in 2005 (Craswell et. al , 2005). The scenario is that given a crawl of the World Wide Web Consortium's web site, a list of candidate experts and a set of topics, the task is to find experts for each of these topics. Following the accessibility to these business corpora, expertise search in enterprise has become an active research area. Craswell et. al (2001); Liu et. al (2005) utilize the standard IR methodologies by first merging all documents related to a candidate expert into a single personal profile prior to the actual retrieval process. On the other hand, probabilistic approaches (Macdonald and Ounis , 2006; Fang and Zhai , 2007) and formal methods (Balog et. al , 2006) have been developed based on the assumption that searching for people is different from searching for documents.

There is limited prior work has been aimed at building real-world publicly accessible expertise search systems for academic institutions. Several faculty expertise search systems have been developed such as Cornell's CALS search³, UIUC's IRIS⁴ and the University of Minnesota's Experts@Minnesota⁵. However, most of these systems are based on Boolean search lacking a notion of a grading scale and thus offer no clue to the relevancy of the results returned in response to a search query. In addition, most of these systems only rely on human-crafted information thus limiting their scope. To ensure high ranking accuracy, we designed a set of novel techniques for effectively combining multiple types of evidence from different information sources, for self-adapting system parameters to various types of queries and for utilizing desirable representation of search queries.

²http://trec.nist.gov/data/t14_enterprise.html

³<http://www.cals.cornell.edu/cals/search/index.cfm>

⁴www.library.uiuc.edu/iris/

⁵<http://experts.umn.edu/>

3. Requirements and challenges

FacFinder is a faculty expertise search engine developed at Purdue university for the INDURE project⁶. The INDURE effort aims at creating a comprehensive online database of all faculty researchers at academic institutions in the state of Indiana. Four universities currently participate in the project including Ball State University, Indiana University, Purdue University and University of Notre Dame,. Together these universities involve over 12,000 faculty and research staff. The participating institutions are encouraged to log into the database to submit the basic information of their faculty such as college, department and research areas. FacFinder is designed to rank these faculty members according to the research expertise related to a user query. For example, a researcher in biology may look for a collaborator with statistics background on protein function classification. The seeker may search for “statistics” in general or “support vector machine” if he/she has a specific technique in mind. In either case, the system should return a relevance-ranked list of faculty with the desired expertise. Specifically, FacFinder needs to support two key requirements: 1) Ranking: give a particular order to the faculty based on the relevance of their expertise and 2) Identification: show the faculty’s profile information such as contact, affiliation and research fields, when searching over their names.

The second requirement is the basic capacity provided by most database systems which should be also covered by more advanced knowledge management systems such as FacFinder. The first requirement leads to an essential functionality in any faculty expertise search system. Some of the challenges to develop an expertise search system are pointed out by [Becerra-Fernandez \(2006\)](#). Beyond these, deploying such a system in an academic environment has additional difficulties. First, manually developing a database containing faculty information is a labor intensive and expensive task. Many faculty members are not willing to devote their time to enter detailed description of their expertise. Secondly, the users of faculty expertise IR systems often have more sophisticated information needs than general search engine users have. When possible, they tend to use complex queries to articulate their requirements. Thirdly, for a project across multiple universities, the scale of such a system is large and efficiency is an important issue for such a real-world application. The system should be able to scale to support a large number of documents to be searched to rank faculty on a given expertise. Other difficulties include maintenance and update of faculty expertise profiles, while minimizing the need for cumbersome, and possibly biased, self-reporting. These challenges involve conflicting design choices faced in the development of the system. The design goal of FacFinder is to achieve a delicate balance between effectiveness, efficiency and flexibility.

4. The FacFinder system

The search process used by the FacFinder software system is illustrated in Figure 4. It consists of a custom CGI script and several indexes from diverse data sources. The search service is provided by a server whose configuration is shown in Table 1. The CGI script is invoked when a user submits (POSTs) a query from the FacFinder web interface. This script opens a TCP/IP connection to the FacFinder server, and at the end formats the results for display to the user. The server accepts requests from the CGI script and executes the core ranking algorithm based on the indexed data. The data currently used come from four data sources: NSF award abstracts (#1), faculty homepage (#2), profile

⁶The retrieved results shown in INDURE combine those of two search engines. The results from FacFinder can be found in the middle of the “Score” section at <http://www2.itap.purdue.edu/indure/search/advanced.cfm>

(#3) and dissertations of their respective Ph.D. students (#4). Among these sources, the latter three are provided by faculty and university libraries while the first is obtained by crawling the Web. The details of FacFinder are discussed in the following subsections.

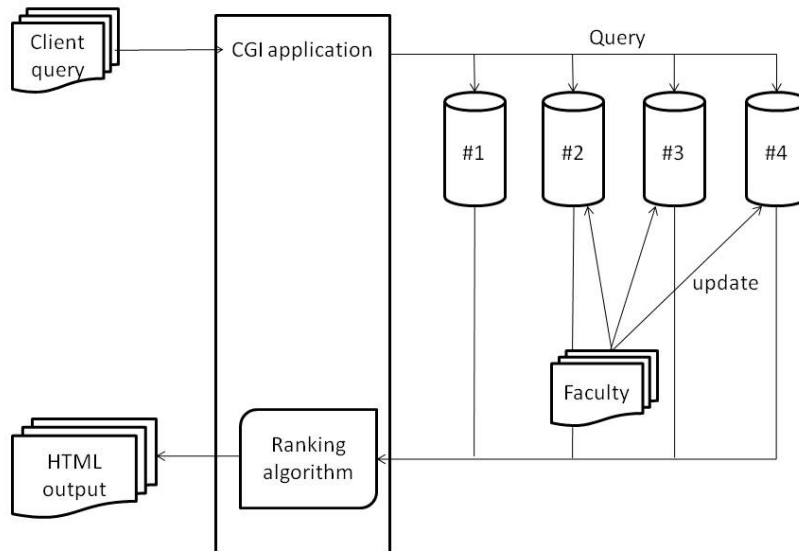


Figure 1: FacFinder search process

Table 1: Machine configuration for the FacFinder server

| | |
|-------------|--------------------------------------------------------|
| CPU | Intel Pentium 4 3.00GHz × 1 |
| Memory | 3GB |
| OS | Linux 2.6.17.10 |
| Bus speed | 800MHz |
| Network | 1Gb/s Ethernet (NetXtreme BCM5751) |
| Boot volume | Western Digital 250GB (WD2500JS), 8.9ms read seek |
| Work volume | Seagate Barracuda 500GB (ST3500630AS), 8.5ms read seek |

4.1. Data acquisition

Data acquisition is the first challenge in developing the system. Initially, the data entered into FacFinder only contains the profiles filled out by individual faculty members and/or their department heads and the abstracts of the dissertations supervised. The profiles include faculty research areas, which could be keywords from a predefined taxonomy⁷ or free keywords that adequately describe the expertise. Some faculty members also provide their personal homepages, which reveal their education, research interests, and/or publication list, and therefore serve as a highly indicative source of their expertise.

⁷<https://www2.itap.purdue.edu/indure/hierarchy.cfm>

The user-authored profiles are inadequate because it would take too long to build a critical mass of information. In addition, the predefined or freely selected keywords are often vague and may not cover faculty's specific research expertise. Therefore, in order to alleviate the data sparsity problem, the documents are stemmed with the Porter stemmer (Porter , 1980). Furthermore, we extracted a large amount of information from the Web. In particular, an initial prototype of the system utilized one external data source: the abstracts of NSF-funded projects. Because of the sensitivity of the data, we follow policies for restricting what data we collect, how we use the data, and what information is made available to users.

4.1.1. NSF-funded projects

The National Science Foundation (NSF) funds research in a wide range of fields of engineering, humanities, and science. The abstracts that describe the research projects that NSF has funded can be found by searching the Award Abstracts database⁸. This database includes both completed and ongoing research and thus provides a valuable source to show some faculty's research expertise. We ran a web crawler to grasp all the abstracts of NSF awards received by faculty with profile in INDURE. The total number of crawled documents is over 60,000. Duplicate abstracts in the database were removed. Duplicate document detection is a difficult problem in certain IR applications, but for the NSF abstracts it is rather straightforward since every funded project has a unique award number contained in the abstract. We note that a significant number of faculty do not get covered in the NSF awards database. These include some junior faculty and the faculty outside of the NSF-supported fields.

4.1.2. Multiple indexes

FacFinder chooses to index full text instead of only the keywords in faculty profiles. This provides the flexibility that a search can be performed using queries that are not traditionally considered expertise terms, such as highly specific technical terms or project names. For information retrieval systems, building indexes is a routine task with mature technology, and many open source software tools provide fast indexing service. For faculty expertise search, there exists extra intricacy: searching for people is different from searching for documents. As discussed in Section 4.1, one faculty member's data are collected from up to four data sources probably ending in multiple documents. It is possible to concatenate the documents to form one single document for each faculty member, but this may bias the ranking toward the data sources with large chunks of text and blur the ones with fewer text but with distinguishing features such as faculty-authored research keywords. Therefore, a FacFinder prototype builds index separately for different data sources and also retrieves the documents from the respective data source. The final ranked list of faculty is obtained by merging and weighting the individually retrieved results. This process is similar to that of distributed information retrieval (Callan , 2000). In fact the algorithm can be implemented to exploit multiple processors thus enabling the system to scale to a large data volume.

4.1.3. Data update

The INDURE database allows faculty members at any time to modify their profile information or add their Ph.D. students' dissertations. Therefore, the data being used and indexed are in constant change. To reflect the latest update, an automatic indexing service is provided in FacFinder, which recurrently updates the indexes. The key ingredient is

⁸<http://www.nsf.gov/awardsearch/>

to set up a process run at regular intervals to check whether the data are modified from the last input by comparing the content of the two data files. If any change is detected, the corresponding index(es) are automatically rebuilt. Because the faculty-authored changes are constrained within the profile and dissertation data, multiple separate indexes here bring the benefit to only update at most two indexes without concern for the others. The homepage and publication data are updated periodically but in much less frequency, because they are rather stable.

4.2. Retrieval and ranking algorithm

Many current faculty expertise search systems are based on two technologies: directories and Boolean search. Directories are like yellow pages which list faculty members by category and subcategory. They require a lot of manual work to classify faculty profiles. On the other hand, the Boolean model is a simple retrieval model based on set theory and Boolean algebra, which is intuitive and easy to understand by naive users, but its retrieval strategy has no notion of a grading scale. A large number of different retrieval approaches have been proposed and studied in the IR research community, and significant effort has been devoted to the evaluation of such approaches, especially in the context of TREC evaluation (Voorhees and Harman, 2001; Craswell et al., 2005; Soboroff and Craswell, 2006). Among them, statistical language models have shown good empirical results. The language modeling approach allows extremely complex queries. This desired property sets FacFinder to lean towards language models as its retrieval strategy, because as discussed in Section 3, the users of the system, when possible, are inclined to explore complex queries to refine their information needs. In particular, FacFinder is built on the Indri⁹ toolbox whose retrieval model in turn is based on language modeling (Ponte and Croft, 1998) and inference network (Turtle and Croft, 1991).

4.2.1. Query transformation

Users typically have poorly articulated requirements in natural language. Therefore, FacFinder translates user queries into Indri's structural query language based on the assumption that query terms are likely to appear in close proximity to each other within relevant documents. For example, given the query "data mining applications," relevant documents will likely contain the terms "data mining" and "applications" which are close to one another in the documents. Many retrieval models ignore the proximity constraints and allow query terms to appear anywhere within a document, even if the words are clearly unrelated. In addition, the order of query terms is important. "Operating system" and "system operating" have completely different meanings. By utilizing the rich Indri query language, the above assumptions can be taken into account. In Indri, #od N (terms) denotes ordered window, i.e., terms must appear ordered, with at most $N - 1$ terms separating them. Similarly, #uw N (terms) means unordered window as all terms must appear within window of length N in any order. With these two operators, term proximity can be encoded into the query. For example, we can specify a relatively small window length for long queries instead of infinity. In addition, the order of query terms can be differentiated by the #od N operator. #weight is another useful operator with which varying weights can be assigned to certain expressions. For example, the exact query phrases need to receive high weights. A similar belief operator is #combine for combining evidence about terms and phrases. To give an idea of how queries get transformed, see the example in Table 2.

By exploiting Indri's rich structural query language, FacFinder provides a set of advanced search options to satisfy sophisticated information needs of users. For example, a user can check the "Research Areas" option¹⁰ prior to

⁹<http://www.lemurproject.org/indri/>

¹⁰<http://www2.itap.purdue.edu/indure/search/advanced.cfm>

Table 2: Translated Indri structural query for “data mining applications”

| |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #weight(0.3 #od1(data mining applications) 0.2 #od30(data mining applications) 0.15 #uw300(data mining applications) 0.1 #od1(data mining) 0.1 #od300(data mining) 0.05 data 0.05 mining 0.05 applications) |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

performing the search. In this case the search will be constrained to the faculty-authored research areas. The wildcard operations are also supported by FacFinder such as the query “nano*” searching for every word starting with “nano.”

4.2.2. Weighting scheme

In FacFinder, the retrieval score $S_{i,j}$ from Indri retrieval model (Strohman, 2005) is not directly merged to form a final score. Instead, the score is transformed by the following exponential functions:

$$S'_{i,j} = \exp((S_{i,j} - \max_i S_{i,j}) \times D_i) \quad (1)$$

where $S_{i,j}$ denotes the i^{th} faculty member’s evidence score from the j^{th} data source. Here, the notation is simplified in the sense that there may exist multiple documents for one person from a single source (for example, several professors will likely have more than one supervised dissertations). In this case, $S_{i,j} = \sum_k S_{i,j,k}$ over k documents. Because most users focus on the top ranked results, we choose the exponential transformation which can make the top ranked documents much more distinguishable from each other than linear function can. D_i is the parameter to control the shape of the exponential function. The final score for the i^{th} faculty member is calculated as

$$f_i = \sum_j \omega_j \times S'_{i,j} \quad (2)$$

where ω_j is the weight associated with the j^{th} data source. Some data sources are more indicative than others such as the profiles the faculty input to the system. Therefore, the profile data should receive high weights. In FacFinder, the values of these weight parameters are empirically tuned to be: $\omega_1 = 3$, $\omega_2 = 4$, $\omega_3 = 6$ and $\omega_4 = 1$.

4.3. Adaptive parameters

Like many other search engines, FacFinder involves several parameters. A common observation is that there exists no parameter values that are maximized for all queries. In other words, some parameter values are good for certain queries while some are inevitably bad for others. These observations drive FacFinder to explore flexible parameters that could adapt to different types of queries.

The prototype FacFinder utilizes simple and intuitive methods to adjust parameters adaptively. Consider an example “computer network”. Some faculty may have many NSF-funded projects related to “computer,” and none of them contains the exact phrase, but a large number of the NSF documents can accumulate the score $S'_{i,j}$ to surpass the faculty who are really in the area. Therefore, we need to prevent the score from collecting many documents within one data source by specifying a threshold γ for the document. For frequently occurring words such as “computer,” the threshold should be high or else many relevant documents will be taken into account. On the other hand, for the rare words, no threshold is needed at all. To avoid trading some queries for others, an intuitive implementation in FacFinder is to specify the threshold adaptively based on the word frequency w such as $\gamma = f(w)$, where $f(\cdot)$ is a monotonically increasing function with respect to w . Specifically, γ takes the following form in FacFinder.

$$\gamma = \begin{cases} S_m & \text{if } n \geq m; \\ S_1 & \text{if } n < m. \end{cases}$$

where S is the sorted array list of the document retrieved scores in descending order and thus S_1 is the first item in the list with the largest score. m is a predefined value characterizing the word frequency and was empirically chosen to be 300 in the system. A large n indicates high occurrence of the word n and therefore γ should be bounded by S_m . Similarly, a small n means low word frequency and therefore no constraint for the word (i.e., γ takes the largest value).

5. FacFinder Evaluation: Methodology

Expert search and traditional ad hoc retrieval are related problems; both attempt to find relevant information items given a query while the information items are people in the case of expert search. Our evaluation strategy for FacFinder closely followed the standard methodology for evaluating ad hoc search systems. The evaluation procedure works as follows: 1) queries are run against the collections; 2) faculty members are retrieved (instead of documents); 3) rankings are judged for relevance to the query.

For ad hoc retrieval, the two basic measures of accuracy are precision and recall. Precision is defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search. Recall is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents. Recall is difficult for use as an evaluation measure for faculty expertise ranking systems because it is difficult to assume how many relevant items there are for a particular query. In fact for the same reason [Gwizdka and Chignell \(1999\)](#) did not include recall in their recommended measures of search engine evaluation. Another measure in Web search is called Normalized Discounted Cumulative Gain (NDCG). The idea underlying NDCG is to allow for multi-level instead of binary relevance. Each relevant web page contributes some “gain” corresponding to the level of relevance. “Cumulative” means to measure the overall utility of documents by the sum of the gain of each relevant document. “Discounted” means to discount the gain of a document ranked low so that a highly ranked document will be “counted” more toward the gain. “Normalized” means to use the ideal ranking to compute a theoretic upper bound of the measure and then normalize the actual gain value with this upper bound. Formally, NDCG is defined as

$$N = M \sum_i (2^{r(i)} - 1) / \log(1 + i). \quad (3)$$

This is a sum over result ranks, where $r(i)$ is the relevance of document in rank position i and M is the normalizing constant chosen so that the score is always between 0 and 1. NDCG is very sensitive to the position of the highest rated people. This characteristic is crucial in expert search, where the majority of the users are mainly focused on the top results. In the experiments, precision ($P@$) and NDCG($N@$) at various document ranks are used to gauge the FacFinder system.

The INDURE project did not already have standard queries for testing the system, so a set of queries on which the evaluators have good judgment were chosen from the query logs of the current system. The queries for the evaluation are listed in alphabetic order in [Table 3](#) (the order of queries does not matter in the experiments). Each query was run against the data collections and the top 20 returned faculty members were investigated. It reflects the real-world situation where the seekers generally do not have time to judge too many faculty members per query. Relevance

assessments were binary (relevant or not relevant) for the Precision metric and triple (excellent, fair, and poor) for NDCG. In order to make the assessments consistent across all the queries, the judgments were completed by one person.

Table 3: Set of test queries with relevance judgments used for evaluating FacFinder.

| | | | |
|-------------------------|----------------------------|-----------------------------|--------------------------|
| Artificial Intelligence | Computational biology | Computer networking | Computer security |
| Data Mining | Human computer interaction | Information retrieval | Linear regression models |
| Numerical algorithms | Machine learning | Natural language processing | Operating system |
| Programming languages | Software engineering | Social network | |

Table 4 shows statistics obtained from the experiments. The table includes the number of faculty, the number of dissertations, the number of faculty-specified homepages, the total number of NSF-funded projects and the number of unique NSF award abstracts (after removing the duplicates). These data contain most of the ones that are currently used in FacFinder, but not exactly the same because FacFinder’s data are updated everyday as discussed in Section 4.1.3. At the time of writing this paper, the dissertation data entering into FacFinder only include those from Purdue University. Therefore, the number of dissertations is relatively small.

Table 4: INDURE database statistics.

| Faculty | Dissertations | Homepages | NSF awards | unique NSF awards |
|---------|---------------|-----------|------------|-------------------|
| 12,327 | 6,946 | 2,503 | 6,248 | 5,873 |

A set of experiments was conducted to evaluate the performance of FacFinder under different scenarios. The scenarios differ from each other in the data included in the system and the number of indexes built. Table 5 documents the scenarios with the corresponding labels.

Table 5: Different scenarios for evaluating FacFinder

| Label | Scenario |
|-------|------------------------------------------------------------------------------------------------|
| S_1 | Only profile and thesis data are used; single index |
| S_2 | Abstract of NSF-funded projects are also included; single index |
| S_3 | All the four data sources are utilized including the faculty-specified homepages; single index |
| S_4 | All the data; multiple indexes with the weighting scheme discussed in Section 4.2.2 |

There are two environments under which the experiments were conducted: normal query and expanded query. The normal query is the Indri query without special transformation of the original user query. For example, the normal query for “data mining applications” is simply “#combine(data mining applications)”. The expanded query is the one with the transformation discussed in Section 4.2.1 and in Table 2.

6. FacFinder Evaluation: Experiments and Results

The goal of these experiments is to address the following questions:

1. How effective is the multiple-index approach by considering the heterogeneities of the different data sources ?

2. Do the expanded queries make a difference in the search results ?
3. How do flexible parameters improve search performance ?
4. How efficient is the search in processing user queries ?

Next we address each of the above questions in light of the data obtained through the experiments.

6.1. Single source of evidence versus multiple types of evidence

In the first experiment we examined the system performance under the four different scenarios: S_1 , S_2 , S_3 and S_4 . The experiment was conducted in the normal query environment. The results are in Table 6.

Table 6: Average precision ($P@$) and NDCG($N@$) at various document rank under different scenarios

| | $P@5$ | $P@10$ | $P@15$ | $P@20$ | $N@5$ | $N@10$ | $N@15$ | $N@20$ |
|-------|-------|--------|--------|--------|-------|--------|--------|--------|
| S_1 | 0.389 | 0.316 | 0.253 | 0.198 | 0.469 | 0.506 | 0.513 | 0.534 |
| S_2 | 0.353 | 0.325 | 0.216 | 0.165 | 0.443 | 0.512 | 0.519 | 0.527 |
| S_3 | 0.412 | 0.381 | 0.339 | 0.227 | 0.472 | 0.523 | 0.527 | 0.536 |
| S_4 | 0.492 | 0.414 | 0.352 | 0.308 | 0.562 | 0.586 | 0.602 | 0.615 |

It is not surprising to see from the table that performance generally improves as more and more data are incorporated. The only exceptions come from S_2 against S_1 . This may be related to the fact that some abstracts contain a chunk of background or introduction content which is not closely tied to faculty’s expertise. A noticeable performance jump can be observed in S_4 against S_3 , which demonstrates the effectiveness of the multiple-index approach.

6.2. Expanded queries

This experiment was conducted in the same scenarios as in the previous section. The difference is that the queries were transformed and expanded. Table 7 presents the results. Generally, these results follow the same pattern as in Table 6 and the conclusions in the previous section were validated. Furthermore, by comparing these two tables, we can see that the query transformation did enhance performance. It seems to play an increasingly important role in retrieval as collections get larger (and probably also noisier). In fact, this observation is consistent with what Metzler et. al (2005) indicated.

Table 7: System performance with expanded queries

| | $P@5$ | $P@10$ | $P@15$ | $P@20$ | $N@5$ | $N@10$ | $N@15$ | $N@20$ |
|-------|-------|--------|--------|--------|-------|--------|--------|--------|
| S_1 | 0.563 | 0.435 | 0.398 | 0.358 | 0.573 | 0.563 | 0.570 | 0.602 |
| S_2 | 0.492 | 0.483 | 0.476 | 0.433 | 0.546 | 0.584 | 0.607 | 0.635 |
| S_3 | 0.591 | 0.510 | 0.508 | 0.460 | 0.597 | 0.607 | 0.631 | 0.688 |
| S_4 | 0.772 | 0.717 | 0.686 | 0.620 | 0.673 | 0.691 | 0.714 | 0.751 |

6.3. System performance with adaptive parameters

In this section, we report the experimental results with adaptive parameters in Table 8. The experiment was conducted in the expanded query environment. By comparing the results to S_3 and S_4 in Table 7, we can see that the adaptive parameters brought reasonable accuracy improvement in some cases. The approach performed well especially with documents at large ranks (e.g. top 20).

Table 8: System performance with adaptive parameters

| | $P@5$ | $P@10$ | $P@15$ | $P@20$ | $N@5$ | $N@10$ | $N@15$ | $N@20$ |
|-------|-------|--------|--------|--------|-------|--------|--------|--------|
| S_3 | 0.595 | 0.518 | 0.512 | 0.471 | 0.599 | 0.610 | 0.639 | 0.698 |
| S_4 | 0.772 | 0.723 | 0.698 | 0.635 | 0.673 | 0.694 | 0.719 | 0.762 |

6.4. Computational efficiency

Since computational efficiency is an issue of interest, the execution time of search is presented in Table 6.4. The execution time is measured as the average elapsed time from submitting a query to retrieving the results. S'_4 in the table denotes the scenario S_4 with adaptive parameters. As shown in the table, the average processing time in all the scenarios is within a second, which is satisfactory for a practical system. Furthermore, the current system only uses a single server. The search efficiency is expected to be substantially improved by distributing the data and search process across multiple servers. Another observation is that the adaptive parameters did not sacrifice computational efficiency by bringing accuracy improvement. Therefore, including the flexible parameters in the search process appears to be a useful approach.

Table 9: Comparison of average running time (T) under different scenarios

| | Normal query | | | | | Expanded query | | | | |
|--------|--------------|-------|-------|-------|--------|----------------|-------|-------|-------|--------|
| | S_1 | S_2 | S_3 | S_4 | S'_4 | S_1 | S_2 | S_3 | S_4 | S'_4 |
| $T(s)$ | 0.57 | 0.64 | 0.68 | 0.75 | 0.75 | 0.63 | 0.68 | 0.71 | 0.78 | 0.78 |

7. Conclusions and Future work

We have introduced FacFinder, a publicly accessible expertise search engine for ranking over 12,000 faculty members across four universities in the state of Indiana. A series of controlled experiments were conducted to demonstrate the effectiveness and efficiency of the system. Preliminary feedback from real users are positive and constructive. One limitation of FacFinder pointed out by users is that it cannot identify synonyms. A typical approach to handling the problem is to expand user queries by using a thesaurus of synonyms. The side effect of this is the loss of precision in the answer due to an increase in the number of retrieved results. How to keep a good balance between precision and recall is worth further exploration.

There are several other promising directions to enhance the system. First, more data can be utilized. For example, faculty publications are a potential valuable data source because of their coverage. Publications are widespread among faculty, and thus the data can be collected for at least research active faculty. Moreover, the number of citations of a publication is a highly indicative factor of the author’s expertise with the publication. The publications can be easily incorporated into the FacFinder framework by building a separate index and assigning weights according to citations. Secondly, some parameters such as ω_j and the D_i in Section 4.2.2 can be learned from the data by supervised learning. Bayesian learning may be particularly appropriate for ω_j because prior knowledge about them is typically available (like profile needs to receive a high weight). Furthermore, the relationship or social network such as co-authorship of

publications and co-PI of NSF awards among faculty members can be exploited. The idea is for faculty to borrow data or information from each other and thus could possibly improve the ranking accuracy. Statistical relational learning techniques (Getoor and Taskar, 2007), which have been proven effective in some cases, can be explored in this direction.

Acknowledgments

We thank the following for their assistance in the INDURE project: Jeffery Vitter for seeding the idea that led to Purdue University Research Expertise (PURE) database, a precursor to INDURE; Kyle Bowen for the early development and continuous support; Jason Fish for his constant hard work in the development of INDURE; Rabindra Mukherjee for funding the early development of PURE; Cindy Nakatsu and Dana Werner for advertising PURE to potential graduate students across the globe; Nate Feltman for believing in the power of INDURE for businesses in the State of Indiana; Brose McVey for serving as a university liaison and keeping us all on track; the able technical staff of Purdue libraries, Sponsored Program Services, Human Resources, and the Office of Technology Commercialization for providing data feeds to INDURE; John Schneider and Jeff Gunsher for testing INDURE and advertising it to the outside world; the many department heads and their able staff who helped enter large amounts of data describing faculty expertise; and last but not the least the upper administration at each of the participating universities for sharing faculty data. The second author is also partially supported by National Science Foundation under the grants 0746830 and No. 0749462.

References

- Balog, K., Azzopardi, L. & de Rijke, M. (2006). Formal models for expert finding in enterprise corpora. In *SIGIR06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 43-50), New York, NY, USA.
- Becerra-Fernandez, I. (2006). Searching for experts on the web: a review of contemporary expertise locator systems. *ACM Transaction on Internet Technology*, 6(4), 333-355.
- Callan, J. (2000). Distributed information retrieval. In *Croft, W. B. (Ed.), Advances in Information Retrieval* (pp. 127-150), Kluwer Academic Publishers.
- Craswell, N., de Vries, A. & Soboroff, I. (2005). Overview of the trec-2005 enterprise track. In *Proceedings of TREC-2005*, Gaithersburg, USA.
- Craswell, N., Hawking, D., Vercoustre, A. & Wilkins, P. (2001). Panoptic expert: Searching for experts not just for documents. In *Ausweb Poster Proceedings*, Queensland, Australia.
- Crowder, R., Hughes, G. & Hall, W. (2002). Approaches to locating expertise using corporate knowledge. *International Journal of Intelligent Systems in Accounting, Finance & Management*, 11(4), 185-200.
- Davenport, T. (1996). Knowledge management at Hewlett Packard, Center for Business Innovation.
- Davenport, T. & Prusak, L. (1998). *Working Knowledge: How Organizations Manage What They Know*, Harvard Business School Press.
- Fang, H. & Zhai, C. (2007). Probabilistic models for expert finding. In *Advances in Information Retrieval, Proceedings of the 29th European Conference on IR Research* (pp. 418-430).
- Getoor, L. & Taskar, B. (2007). *An Introduction to Statistical Relational Learning*, MIT Press.
- Gwizdka, J., & Chignell, M. (1999). Towards information retrieval measures for evaluation of Web search engines. Retrieved from http://www.imedia.mie.utoronto.ca/~jacekg/pubs/webIR_eval1_99.pdf
- Lin, C., Griffiths-Fisher, V., Ehrlich, K. & Desforges, C. (2008). SmallBlue: People Mining for Expertise Search and Social Network Analysis, *IEEE Multimedia Magazine*.
- Liu, X., Croft, W. B., & Koll, M. (2005). Finding experts in community-based question-answering services. In *CIKM05: Proceedings of the 14th ACM International Conference on Information and Knowledge Management* (pp. 315-316), New York, NY, USA.
- Macdonald, C. & Ounis, I. (2006). Voting for candidates: adapting data fusion techniques for an expert search task. In *CIKM06: Proceedings of the 15th ACM International Conference on Information and Knowledge Management* (pp. 387-396).
- Maybury, M., (2006). Expert Finding Systems. *Technical Report MTR06B000040*, MITRE Corporation.
- Metzler, D., Strohman, T., Zhou, Y., & Croft, W.B. (2005). Indri at TREC 2005: Terabyte Track, In *Proceedings of 2005 Text REtrieval Conference*.
- Ponte, J. M. & Croft, W. B. (1998). A language modeling approach to information retrieval, In *Proceedings of the 21st Annual international ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 275-281).
- Hertzum, M. & Pejtersen, A.M. (2000). The information-seeking practices of engineers: searching for documents as well as for people. *Information Processing and Management*, 36(5), pp. 761-778.
- Porter, M. F. (1980). An algorithm for suffix stripping. In *Program: Electronic Library and Information Systems*, 14(3), 130-137.

- Soboroff, I., de Vries, A. P. & Craswell, N. (2006). Overview of the TREC-2006 Enterprise Track. In *Proceedings of TREC 2006*.
- Strohman, T. Metzler, D. Turtle, H. & Croft, W. B. (2005). Indri: A language model based search engine for complex queries. In *International Conference on Intelligence Analysis*.
- Turtle, H. & Croft, W.B. (1991). Evaluation of an Inference Network-Based Retrieval Model. *ACM Transactions on Information System*, (pp. 187-222).
- Voorhees, E. & Harman, D. (2001). In the Proceedings of Text REtrieval Conference (TREC1-9). *NIST Special Publications*. <http://trec.nist.gov/pubs.html>.
- Wright, A. & Spencer, W. (1999). The National Security Agency (NSA) networked knowledge and skills management system. In *Proceedings of Delphi's International Knowledge Management Summit*.
- Yimam-Seid, D. & Kobsa, A. (2003). Expert finding systems for organizations: Problem and domain analysis and the DEMOIR approach. *Journal of Organizational Computing and Electronic Commerce*, 13(1), pp. 1-24.