KTH Computer Science
and Communication

Determining expertise from indirect evidence

Expert search in an enterprise setting

VIKTOR HOLMBERG
<VHO@KTH.SE>

Master's Thesis at NADA
Supervisor: Johan Boye <jboye@csc.kth.se>
Examiner: Jens Lagergren <jensl@csc.kth.se>
Project provider: Simon Stenström <simon.stenstrom@findwise.com>

TRITA xxx yyyy-nn

Abstract

In large organizations, quickly finding the right expert is important. An expert finding system can help users with this by letting them search for experts matching expertise queries. *Automatic* expert finders, which are the focus of this report, accomplish this by analyzing indirect expertise evidence contained in documents, for instance those stored on an organization's internal network. The goal of this report is to discover the best way to construct such a system, and analyze how its performance compares to that of a system based on manually supplied expertise information.

To determine this, several expert finding algorithms were implemented, and evaluated on two different corpora. It was found that the best approach, both in terms of performance and ease of implementation, is to use document-based algorithms, i.e algorithms that work by first searching for documents related to a query, and then analyzing the retrieved documents to find presumably relevant persons. Furthermore, the best of the implemented expert finders was compared to an existing manually curated system. The automated expert finder outperformed the manual one substantially in terms of recall, while maintaining high precision.

It was concluded that automatic expert finders perform well enough to be useful in practice, and that the workload needed to implement such a system is small. As such, they can provide either a feasible alternative, or a valuable complement, to manual systems.

Sammanfattning

Att avgöra expertis utifrån indirekta informationskällor

I stora organisationer är det viktigt att snabbt kunna hitta rätt expert. Ett sätt att hjälpa användare med detta är expertsökarsystem, vilka låter användare söka efter personer matchande en given expertis. I fokus för denna rapport är *automatiska* expertsökarsystem, vilka bygger upp expertismodeller från indirekta källor, så som dokument lagrade på ett intranät. Vi undersöker hur ett sådant system bäst konstrueras, samt hur dess prestanda förhåller sig jämfört med ett system baserat på manuellt skapad expertisinformation.

För att utröna detta implementerades ett antal olika expertsökare vars prestanda testades på två olika dataset. Det visade sig att s.k dokumentbaserade algoritmer fungerar bäst, både med hänsyn till kvalitén på sökresultaten och till mängden arbete som krävs för implementation av systemet. I en dokumentbaserad expertsökare används en klassisk dokumentsökmotor för att skapa en lista av dokument relevanta för en expertissökfrågan, varpå dessa dokument analyseras för att hitta potentiella experter.

Den bästa av de implementerade expertsökarna jämfördes med ett existerande system baserat på manuellt ifylld expertisinformation. Det visade sig att det automatiska systemet kunde lokalisera en betydligt större andel av de faktiska experterna än det manuella systemet, med bibehållen hög precision.

Vi drar slutsatsen att automatiska expertsökarsystem fungerar bra i praktiken, samtidigt som arbetsbördan för implementationen av ett sådant system är liten. Således kan de fungera antingen som ett värdefullt komplement eller som ett realistiskt alternativ till manuella system.

Contents

Chapter 1

Introduction

In large organizations, especially knowledge intensive ones, knowing *who knows what* is often as important as it is difficult. It is important because being able to quickly locate an expert on a topic can save time, energy and money. It is hard because expertise is often not documented, and even when it is, the documentation is rarely as specific or complete as one would like.

Also, large organizations often have a lot of data in the form of various documents stored on an internal network. What if we could use all this data to solve the problem above?

This is precisely the goal of an *automatic expert finder*. By analyzing data obtained from a document repository such as an internal network, it can provide answers to questions of the type "Who knows about X?". Such systems have the potential to offer valuable support for a person seeking an expert, without requiring manual labor to maintain expertise information on all the individuals in the organization.

In this study, we seek to answer the following questions:

- What is the best[1] way to automatically locate experts in an enterprise setting?
- How does this approach compare to manually administered systems for determining expertise?

To answer these questions, we have implemented a number of automatic expert finders, and compared their performance on two different datasets. Their performance was also compared against a system based on manually supplied expertise information.

[1] We will elaborate on what we mean by *best* later, in section 2.3. For now let us just say that we will look primarily at objective retrieval quality metrics, but also take into consideration other factors such as difficulty of implementation.

1.1 Definitions

In order to avoid confusion, definitions of important terms used throughout the report are given here.

Expert finder A system that helps users find experts with a given expertise. Usually, the input of the system is an expertise query, and the output a list of matching experts.

Expert profiling The problem of describing the expertise of an expert. That is, for a given person, return a human readable description of his or her expertise.

(Expert) seeker The user of an expert finder system, looking for experts on a topic. In this report, the term *seeker* will always mean *expert seeker*, unless otherwise specified.

Candidate A person who might be an expert. For example, in a corporation we would assume every employee is a candidate in the expert finder system.

Document In this report, we take the term document to mean a single "piece of text". Examples include web pages, word documents, emails, or source code.

Candidate profile An explicit model of a user created by an automatic expert finder. In general, it is not easily readable to humans. As such, just the fact that the expert finder creates candidate profiles does not mean we have solved the expert profiling problem mentioned above.

Information need A query posed to a document or expert search system. Also called *test topic*, and usually implies that an associated list of desired results is available for evaluation.

Personal page In this report, a personal page is a manually created profile of a user in human readable format. For example, a web page containing free text about the person, or a collection of skill keywords.

Explicit evidence Evidence that is clearly and precisely presented. In the context of this study, explicit evidence is data created with the intension of conveying expertise.

Implicit evidence "Hidden" evidence that implies or suggest a conclusion, but is not readily presentable. In this study, the conclusion in question is usually the expertise of a candidate.

Chapter 2

Problem background

In this part of the report, motivations for working on the expert finding problem will be given, along with high-level descriptions of different approaches to solving it. Also, some desirable properties of an expert finder system will be listed.

2.1 Reasons for needing an expert

As mentioned in the introduction, finding experts on certain topics is of high importance in large organizations [13]. The more knowledge intensive and diverse the tasks of individuals within the organization are, the more critical the issue of locating experts becomes. The question of how to find experts is by no means a new one, but its importance has grown along with an increasingly knowledge-intensive and specialized work environment in recent years.

Some reasons for needing an expert might be:

- **Need to access non-documented information**
  It is not possible for any organization to explicitly document all the information available to it. No matter how rigorously knowledge is documented, it will still be the case that certain information can only be effectively transferred through apprenticeship, experience and informal conversation. Such experience that is hard to document includes novel situations, work in progress or general knowledge. [12]

  The importance of accessing non-documented information is obviously intensified when documentation is lacking. This is common in practice, for instance in the software engineering industry [10].

- **Seeking improved efficiency**
  Sometimes, what is a lot of hard work for a non-expert might be very little work for an expert. Time and energy can be saved if people within an organization can help each other out on difficult tasks, or swap tasks if they find that the assignment of tasks was non-optimal. For instance, two persons might be experts in two different fields, but have been assigned tasks from each other's

fields. If they could simply figure out their respective expertise, they could potentially cut down the work needed to accomplish their collective workload significantly by swapping tasks, leaving them both to focus on more fruitful endeavors. [26]

- **Need for specification**
  A user might have only a vague idea of a problem he needs to solve. The right expert can be of key importance in aiding the user in precisely identifying the problem. For example, the solution to a user's problem might already be documented and easily located on an internal network. Even so, this does not help the user if he does not know the relevant terms to look for. For example, the user might not know what queries to give a document search system. The aid of an expert could in this case dramatically reduce the time and energy required to find the solution. [16]

- **Seeking an expert to perform a given function**
  An expert seeker might need to find experts for performing specific functions. Examples of such functions could be internal business roles, e.g. team members for a project, collaborators etc. It could also be external business roles, e.g. an expert as a consultant, employee or contractor.

  In these settings, the search for experts is usually more thorough than in the previous examples, and might demand more precise information about the experts. It is, for instance, often desirable for the seeker to obtain a list of experts ranked by expertise, instead of just an unordered set of possible experts. [26]

- **Socialization need** Users might prefer the human dimension that is involved when asking an expert for help, as opposed to interacting only with computers and documents. [26]

2.2 Expert finding systems

In the preceding section, some scenarios where individuals within an organization need assistance locating experts were outlined. We will now detail how computer systems can provide assistance in such scenarios. The expert finding systems will be classified according to who creates the basis for the expert search; in *Expert databases* the work is done centrally by a group of people with special knowledge on who knows what. In *Personal pages* the work is distributed over all the individuals in the organization. Finally, in *Automatic expert finders* the task of figuring out who has what expertise is left to computers.

Expert databases One of the first computer-aided ways to provide assistance for locating experts developed was *expert databases*, in which a central authority creates expertise models of candidates by manually entering data [26]. Although

the simplicity of this approach lends it some virtue, it suffers from a number of shortcomings:

- Manually creating a database is time consuming and expensive [26].
- Expertise and skills can change quickly, especially environment and organization-specific skills. This will cause a manually created database to quickly become outdated, and as a result more time and resources must be spent updating it. [26]
- Experience and skill descriptions will usually be general and incomplete. However, the queries given to the system will usually be specific and fine-grained. Bridging this gap will usually be the task of the user, who has to make his query broader, and then figure out the specific knowledge set of the matching experts himself. [15]

Personal pages A more modern but similar approach is *Personal pages*, commonly personal *web* pages or similar, on which individuals within the organization provide information on themselves and their expertise. These can then be searched using conventional document search methods. If the individuals within the organization take the time to maintain up-to-date information on their personal pages, a search of these pages will hopefully give the user seeking experts relevant results.

The main advantage of using this approach is that the individuals themselves can usually provide more extensive and precise information on their own skills than an outside person could, especially if the organizational distance between the person assessing the skill and the candidate is large.

Another advantage of this approach is that the data entered is usually in a human-readable format, which might not be the case for an expert database. This can benefit the expert seeker who will potentially overview many expert profiles. This might also make users more willing to keep their profiles up to date and detailed. However, all the fundamental flaws of expert databases listed earlier will still be present. [26]

The distinction between expert databases and personal pages is not always clear-cut. In practice, many manually curated systems feature aspects of both. For instance, instead of full text personal pages, the personal pages might consist of an employee name and contact information, plus skill keywords selected by the employee from a set of predefined skills.

Automatic expert finders The drawbacks of manually managed expert search systems have prompted research into fully or partially automatic solutions to the problem of locating experts.

Modern organizations usually have a large amount of digital data in the form of documents on internal or external networks, such as e-mail correspondence, memos, employee records etc. Automatic approaches to the expert finding problem strive to extract expertise information from these data repositories. This eliminates, or

at least decreases, the need for humans to supply the information directly by obtaining it indirectly from second hand sources that are already present within the organization.

As such, a well working automatic expert finding system has the potential to cut down on expensive human labor; either for the personnel in charge of creating the expert profiles, or for the end users whose time spent searching for experts can be cut down.

2.3 Desirable properties of an automatic expert finder

In this report, we focus on automatic expert finders, such as described in section 2.2. In order to evaluate such a system, we must first determine what makes it useful. In this section, some desirable properties of an expert finder will be listed. These properties will be classified according to whether we hold a end user or administrator/developer perspective.

Note that some metrics are directly measurable, while others are more subjective and hard to measure. In this report, we will focus mostly on the former. However, the more subjective aspects must also be taken into consideration in order to form a complete picture.

2.3.1 User perspective

Relevance Obviously, the experts a seeker is presented with must be relevant to the topic at hand. This is closely related to the standard IR measurement *precision*. Note that there could be other considerations apart from simply the skill levels of the candidates. For example, lets say a corporation has two offices, one in Sweden and one in the United States. A seeker in the Swedish office might be more interested in experts also in his or her own office, even though there could be a more qualified expert in the US. This kind of relevance is referred to as *organizational* or *social* relevance.

Recall A good expert finding system should have a high recall, that is, give the expert seeker a complete list of the relevant experts, or at least as complete as possible. This can be useful in situations where organizational or social relevance trumps pure expertise. In fact, if might be a common situation that the seeker does not need the absolutely best expert, and can be satisfied with an expert who has just a rudimentary knowledge of the topic at hand. In these cases, it is desirable that the seeker can review a large number of experts, and pick one that is organizationally or socially close to the seeker.

Another situation when high recall is needed is when the expert seeker is looking to put together a team of people with similar skills to solve a particular problem. In this case, the seeker would obviously not be satisfied with finding only one expert.

Yet another advantage of having high recall is that if more experts can be found, the user can compensate for false positives, that is, candidates that the system falsely recognized as experts when they did not actually have the expertise needed.

Analysis support We must not forget that in the end, it is the user who selects experts. Even the best of systems can only support the user in this process. For example, it is desirable that the seeker does not get just a plain list of names, but also some information about the experts, such as a short description of each expert [26]. One way to do this is by automatically creating human readable profiles for each expert, *expert profiling*. However, solving the expert profiling problem is out of the scope of this study.

Another way to assist the user is by increasing the transparency of the system. For example, if the expert search results are based on documents, the user can be presented with the documents that made the system recognize an expert as such. [26]

2.3.2 Developer/ administrator perspective

The developers and administrators of an expert finding system might have different views of what makes a good expert finder, compared to the users. As such, it is useful to specify some points explicitly. Of course, normal software development principles still apply, and these will not be detailed.

Ease of implementation The workload needed to implement and maintain an expert finding system should be as small as possible. As such, it is advantageous if the system can be easily deployed alongside or on top of existing information retrieval infrastructure, reusing as many parts of existing systems as possible. Such reusable parts might include document indexing, end-user interfaces etc.

Generalizability The expert finding system should preferably work well without relying on collection specific heuristics or a priori judgments about the data. Examples of such would be to manually give more importance to certain documents, or exclude some. If the amount of such optimizations is small, the same system can be deployed in different organizations or settings with minimal overhead. Relying on such optimizations, on the other hand, will generate a lot of work when adapting the system to a new environment, as they must be re-done to work in a new context.

Chapter 3

Theoretical background

In this chapter, a model for analyzing and describing expert finders is introduced. We will also give an overview of some existing expert finding systems.

3.1 Domain model

As one can imagine, there are many ways to construct an expert finder. This section will therefore introduce a *domain model* of expert finders, consisting of different *domain factors*, each representing a conceptual part of an expert finder. The model used in this report was introduced by Yimam-Seid and Kobsa, and consists of the following domain factors [26]:

- Basis for expertise recognition
- Expertise indicator extraction
- Expertise modeling
- Query mechanisms
- Matching operations
- Output presentation
- Adaptation and learning operations

Basis for expertise recognition The basis of expertise recognition denotes the data, or *evidence*, used to determine expertise. It can be separated into explicit and implicit evidence. Explicit evidence is very useful when it exists, however the focus of an automatic expert finding system is to utilize implicit evidence, for reasons we detailed in section 2.2. Examples of implicit evidence could be document authorship, the documents themselves, e-mail data or patterns, queries to an information retrieval system, data in a structured employee database etc. In theory, virtually any kind of data can be used, since data in an organization is

always created, at least implicitly, by someone and about something. However, for practical reasons, usually only a subset of the evidence available can be used. [26]

Expertise indicator extraction The *Expertise Indicator Extraction* denotes how *expertise indicators* are extracted from the *basis for expertise recognition*. Here, a separation can be made between *Domain knowledge independent* or *Domain knowledge driven* expertise indicator extraction. In the *Domain knowledge independent* case, no assumptions about the data are made. In a *domain knowledge driven* model on the other hand, some knowledge about the data is used to extract the indicators. An example would be using knowledge about the structure of software code documentation to extract experience indicators, as is done with Java code documentation by Vivacqua [25]. Another more basic approach would be to use the author metadata of documents.

Expertise modeling Based on the expertise indicators, *Expertise Modeling* creates an expertise model. This model is not the same as say, entries in an experience database, since it involves an uncertainty concerning how well it matches the real world.

Experience models can be generated at an initial processing of the evidence, or at query time. In the second case, the model might not even exist as a specific data structure. It could simply be a step in an algorithm; in this case, the expertise model is an assumption or an idea embodied in the algorithm.

Query mechanisms The *query mechanism* describes how the expertise queries are generated. The system can require an explicit query from the user seeking an expert, or infer this from user behavior. An example of obtaining queries indirectly would be to give the user contact information of an expert automatically when he spends more than 5 minutes looking at a certain help page for a system.

Matching operations The *matching operations* of the system determine how a query is matched towards the expertise model to obtain experts. If the expertise model is simply text describing each expert, this can be done using simple keyword matching, vector space models or another method entirely.

Output presentation A basic example of output is a ranked list of experts in order of relevance. Sometimes, the experts might not even be ranked. In other scenarios it might be desirable to rank the experts on different criteria (multidimensional ranking), or use an altogether different way of presenting the results.

Furthermore, other information can also be presented to the user, providing analysis support. This might include relevant details about each expert, the context that made the system recognize a person as an expert, social context etc.

Adaptation and learning operations The results of the expert finder might be customized and refined using relevance feedback. For instance, the system might ask seekers to rate the relevance of the experts returned after a search has been completed, and use this new evidence to refine future output. This could be done globally for all future similar queries, or the adaption might be personal for just the current user. This could also be done indirectly; for instance the system might use a seekers past queries to create user models, which could then be compared to those of experts to create tailored results.

3.2 Expert finding algorithms

Balog et al. identified that most automatic expert finding algorithms fall into one of two main types, depending on their expertise model: profile-based or document-based. [3]

Profile-based algorithms In profile-based expert finders, an explicit model of each candidate is created at indexing time. At query time, the query is compared to the profile of each candidate in order to find the best match; see figure 3.1. Different variants of profile-based searchers can differ in, for instance, how the profiles are created, or how the matching from query to profile is made.

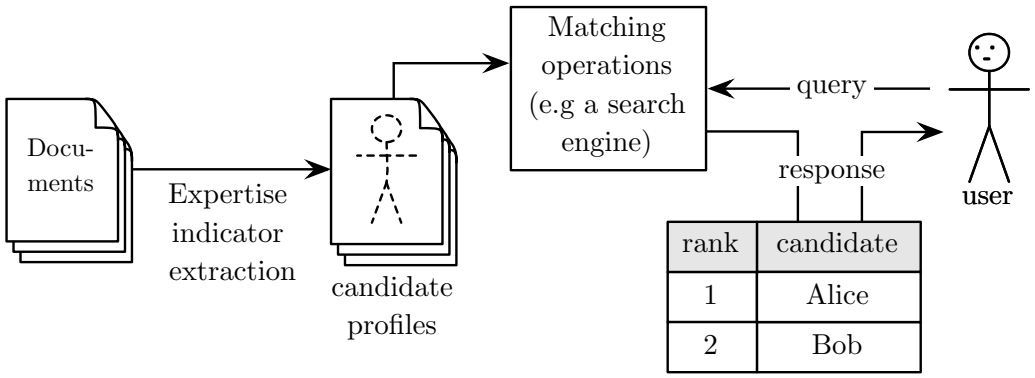


Figure 3.1: Diagram of a profile-based expert finding system.

The creation of the profiles for each user is obviously a critical step for algorithms of this family. One straightforward way of accomplish this is to collect all the documents in which a candidate figures, and combine these into one big "super-document" serving as the profile of a candidate.

Other ways to generate the profiles might incorporate domain knowledge, for instance by weighting some documents more than others when creating the profile. Also, proximity based methods might be used, where terms occurring close to a person occurrence in text are weighted more heavily than far away terms. Window

or field based methods are a variant of this, where only text occurring within a set distance from the query are considered.

The matching of search queries to profiles is usually accomplished using standard text retrieval methods, like vector space similarity.

Document-based algorithms Document-based methods emulate the way a human expert seeker might identify experts when he only has access to a document search engine and not a dedicated expert finding system. In such a situation, the expert seeker might search for documents related to his query, then skim through the retrieved documents. He can then infer that candidates that are mentioned in relevant documents are likely experts. If the seeker is thorough enough, he could potentially get a good estimate of the skill level of different candidates.

As such, document-based expert finders do not construct any explicit models of expertise. Instead, a conventional document retrieval system is used to find documents for a query. Then, the documents retrieved are analyzed in some way to obtain a list of experts. For example, experts who are featured in the most relevant documents can be considered the most relevant experts, while candidates not mentioned in any relevant document can be deemed irrelevant. A schematic view of a document-based expert finder and surrounding systems is shown in figure 3.2.

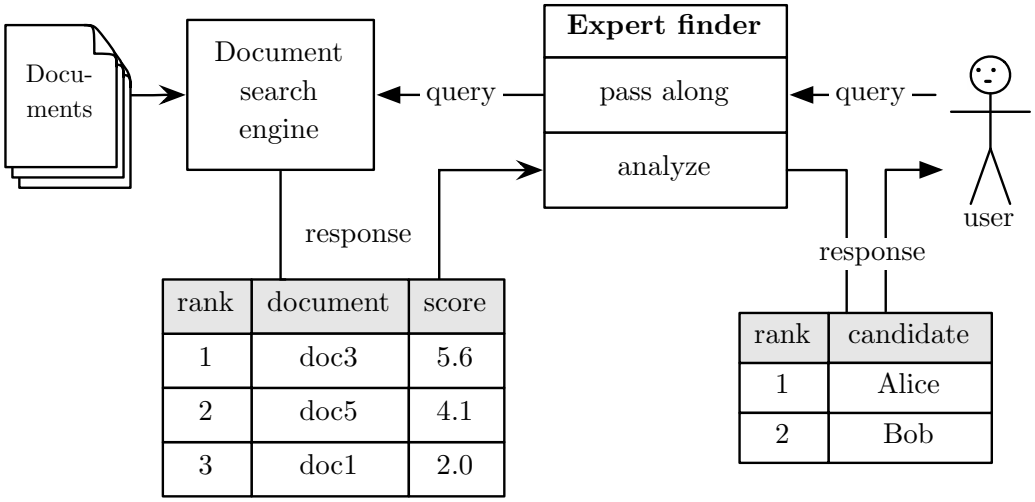


Figure 3.2: Diagram of a document-based expert finding system.

3.3 Related work

A major forum for the recent development of expert finding system has been the TREC (*Text REtrieval Conference*) enterprise track. The TREC enterprise track ran from 2004 to 2008, and included an expert search task. Supplying a corpus

and test topics, it provides a common platform for researchers to empirically asses the performance of different techniques. As a result, the TREC enterprise track has been the basis for evaluating performance of expert finding systems in much of recent literature. TREC has a contest format, so for each year winners are selected based on retrieval performance. [3] [19] [17]

The TREC enterprise track employed one corpus for the 2005-2006 contest, and a different one for 2006-2007. Since the final TREC 2008 enterprise track can be expected to feature the most advanced algorithms, the top three submissions for that year will be detailed. However, for this study, only the corpus used in 2005 and 2006 was used for evaluation.[1] Therefore, the 2005 and 2006 winners will also be described.

TREC 2005 The winner in the TREC 2005 task was **THUENT0505** by Fu et al. [9]. It employed a profile-based method, collecting and combining information about each candidate into a candidate profile. For this reason they called their method *document reorganization*. They also exploited the fact that the test topics for 2005 was working groups with the members of the groups as the desired results. Aiming to establish working group - candidate associations, they specially tailored their algorithm to working group pages, which was labeled as such in the corpus. [11][3]

TREC 2006 The 2006 top submission was **kmiZhu1**, a document-based method by Zhu et al. [24]. PageRank was employed to establish document authorities, which were then considered in the query-document matching step. The document-candidate association model was refined by using the HTML structure of documents to match them against several predefined types such as technical reports, emails or research papers. Further refinement of the document-candidate model was achieved by exploiting query-candidate proximity. [28]

TREC 2008 The top scorers of the TREC 2008 expert search task were: [6]

1. **Balog et al.** used an ensemble method, consisting of both a profile and a document-based approach. [4]

   The profile-based method used a window-based approach on web data obtained from searching the external web for the candidate name; text close to candidate occurrences in the results was collected into the profile for each candidate. The query-profile matching employed a language modeling retrieval model. A much simpler precursor to this profile-based algorithm was implemented for the purposes of this study, as described in section 4.1.1.

[1] The TREC enterprise track's 2007-2008 corpus had similar characteristics to the 2005-2006 one. The reason for not using the 2007-2008 corpus was that the legal requirements for obtaining this corpus was stricter.

The document-based method uses a language modeling-based document retrieval system to obtain a list of relevant documents. Then, for each candidate ca, the sum

$$\mathrm{P}(q|\,ca) = \sum_{d \in D} P(q|\,d) \cdot P(d|\,ca)$$

is calculated, where D is the set of retrieved documents for query q, $P(q|\,d)$ is equivalent to the document score, and $P(d|\,ca)$ is modeled by a document-candidate relatedness measure. The candidates with the highest values for this sum are assumed to be the most relevant to the query. [4]

The final result was calculated as a linear combination of the profile-based method with weight 0.25, and the document-based method with weight 0.75. [4]

2. **Shen et al.** used a profile-based model, where authoritative candidates were weighted higher than un-authoritative ones. The candidate authorities were obtained by constructing a network of persons from co-occurrences in the text, and applying the PageRank algorithm to this network. [23]

3. **He et al.** used a voting model, where documents relevant to a search query are considered as votes for persons related to those documents. Adapting ideas from data fusion techniques, candidates were rewarded for high scoring documents and also for being present in many retrieved documents. This method was further refined to consider query term - candidate proximity in the documents. Also, candidate normalization was used so that candidates occurring in a large number of the documents in the full corpus were given less weight per occurrence than candidates occurring in few documents. [14]

Chapter 4

Method

In this section, we will describe the algorithms implemented, and why these were chosen. We will also describe the datasets and the measures used for evaluation.

4.1 Implemented algorithms

For the purposes of this study, one profile-based and several document-based algorithms were implemented and evaluated. The selection of algorithms was made with the intent of striking a balance between high performance, as evidenced by literature, and ease of implementation. Furthermore, domain specific heuristics were avoided as far as possible.

In all the algorithms presented, implicit evidence in the form of documents is used as the basis for expertise recognition. There is no adaptation or learning with regards to user behavior. The output of the algorithms is simply a ranked list of candidate names; we do not consider more intricate ways to present the output nor the design of any accompanying GUI. For all algorithms, the assumption is made that a set of associated candidates for each document is available; the creation of these lists will be detailed in section 4.2.

4.1.1 Profile-based algorithm

In the *profiles* algorithm, the candidate profile of each user is created at indexing time by including all the terms in all of the documents in which the candidate figures, creating a "super-document" in the same way described in section 3.2.

To match queries to the profiles we used a TF-IDF vector space model.

4.1.2 Document-based algorithms

As described in section 3.2, a document-based expert finder works by first retrieving a list of documents related to a query, and then analyzing this list.

In this study, seven different ways of analyzing the retrieved document list were implemented. The input for all of the different *analyzers* is the same: a ranked

list of documents, where each document is associated with a list of related experts and a number indicating its ranked retrieval score. The scores of the documents, and consequently the ranking, is produced by the document search engine; for this study, the scores were produced using a TF-IDF similarity measure.

In order to improve the speed of the system, we cut off the document list after a set number of documents have been retrieved. This number was set to 200, as this corresponded approximately to each query taking less than half a second in the experimental environment. Empirical tests showed that adding more documents did not improve the results of the expert finders noticeably.

We will now present the analyzers implemented. For a summary of their respective formulas see table 4.1.

Ranks In the *ranks* analyzer, we take the simple approach of ranking candidates in the order they appear in the document list. If multiple candidates are associated with a document, they are ranked in arbitrary order.

Votes Consider each document as a single vote for the candidates associated with it. The *Votes* analyzer ranks the candidates according to who got the most such votes.

CombSUM For all the candidates, sum the document retrieval scores for each document associated with the candidate. *CombSUM* ranks candidates according to this score.

ExpSUM For all the candidates, form the sum of the exponentials of each document score, and use this sum as the basis of ranking. This gives more weight to high-scoring documents.

Analyzer	Score for candidate C
Ranks	n/a
Votes	$\|D_c\|$ or $\sum_{d \in D_c} 1$
CombSUM	$\sum_{d \in D_c} S(d)$
ExpSUM	$\sum_{d \in D_c} e^{S(d)}$
ExpMNZ	$\|D_c\| \times \sum_{d \in D_c} e^{S(d)}$
NormExpSUM	$\sum_{d \in D_c} e^{\frac{S(d)-\min(S)}{\max(S)}} - 1$
NormExpMNZ	$\|D_c\| \times \sum_{d \in D_c} e^{\frac{S(d)-\min(S)}{\max(S)}} - 1$

Table 4.1: Summary of implemented algorithms. D_c is the intersection of documents retrieved for a query and documents associated with a candidate c. $S(d)$ is the score of a document d.

ExpMNZ takes the same sums as in *ExpSUM* and *Votes*, and for each candidate multiplies them together. Thus, *ExpMNZ* rewards candidates that have both

highly scored documents and a large number of documents associated with them. A descendant of the *ExpMNZ* analyzer was used by He et al. in their TREC 2008 submission, which scored third best [14]. This expert finder was described in section 3.3.

NormExpSUM This analyzer is, as the name suggests, a modified version of the ExpSUM analyzer. We normalize the document scores before applying the exponential function, so that for each query the first document in the cut-off document list gets a score of 1, and the last document gets score 0, i.e.:

$$score_d{}' = \frac{score_d - \min(scores)}{\max(scores)}$$

Notice that if the scores before normalization fall in a small interval in the cut-off document list, the impact of this normalization increases. In this way, we make sure that the scores of documents in the cut-off list become spread across the same interval ($[0, 1]$), regardless of their prior spread.

Another aim of this normalization is to ensure that regardless of the query-document retrieval mechanism (some scoring models might give higher or lower document scores in general than others) the response of the exponential function is similar.

In addition to the normalization of document scores, we also subtract 1 from the score addition contributed by each document after the exponential function has been applied. Without this step, documents with score 0 still contribute to the candidate scores, as $e^0 = 1$. As such, subtracting 1 from the score additions increases the boosting of high-scoring documents, and makes sure that irrelevant documents do not affect the candidate ranking.

To the best of our knowledge, these two proposed improvements to the ExpSUM algorithm have not been investigated in literature.

NormExpMNZ is, as the name implies, the same as ExpMNZ but with the same normalization and subtraction of 1 from the per-document score addition as in NormExpSUM.

Choice of analyzers Obviously, there is a large number of ways to analyze the retrieved document list, out of which only a small subset was implemented for this study. The reasons for using the analyzers were:

- Ranks, Votes and CombSUM were included as baseline models.
- ExpSUM and ExpMNZ was chosen because earlier research, such as the work by MacDonald et al. indicate these as being high performing. [19]

- NormExpSUM and NormExpMNZ are novel attempts to improve the performance of the ExpSUM and ExpMNZ techniques.

Note on method names The names of the methods was chosen to be aligned with literature such as MacDonald et al.[19], who in turn took the names of corresponding data fusion techniques, first introduced by Fox and Shaw [7]. The *MNZ* suffix in ExpMNZ and NormExpMNZ stands for *Multiply by number Non-Zero* [2]. Note that we slightly diverged from the naming convention in data fusion literature to allow for easier reading: ExpSUM would usually be called ExpCombSUM, ExpMNZ called ExpCombMNZ, and so on.

4.2 Experimental setting

To ascertain the effectiveness of the expert finding systems designed, their performance was evaluated on two different datasets; the W3C corpus used in the TREC 2005-2006 enterprise track, and a subset of the internal network of Findwise AB, a Swedish IT-consulting company. Using the W3C dataset allows performance comparison with state of the art expert finding systems; the Findwise dataset lets us see how expert search systems can be integrated into pre-existing enterprise search infrastructure. Since Findwise already had a manual expert search system in place, usage of this corpora will allow us to see how such manually curated systems compare to automatic ones. Furthermore, usage of both corpora will allow us to see whether or not the same relative performance amongst the different algorithms remains constant across datasets with different characteristics.

In what follows we will present the nature of these corpora, how the documents and candidates were linked together, and how test topics and ground truth was obtained. A short overview of the corpora properties can be found in table 4.2.

For both corpora, the document search and the profile-based searcher uses Apache Solr for indexing and searching. The similarity measure used for ranking was Solr's default TF-IDF-based similarity measure [18].

Dataset	W3C 05	W3C 06	Findwise
Documents	331,037		676
Candidates	1092		98
Candidates located by	In-text occurrences		In-text occurrences and author tags
Test topics	50	54	29
Relevant candidates per query (median)	18	50	8

Table 4.2: Dataset comparison

4.2.1 W3C dataset

In the TREC 2005-2006 enterprise track, the corpus used was a crawl of the W3C website, containing technical documents, mailing lists, source code, a wiki, personal homepages and more. In total, the corpus contains some 331,037 documents. Also included in the data is a list of all 1092 candidates.

In this study, both the 2005 and 2006 test topics were used. For 2005, a ground truth was obtained from the working groups of W3C, taking the names of the groups as topics and the members of the groups as desired results. In 2006, a survey was conducted to obtain a ground truth.

We used a version of the corpus with candidate occurrences already tagged by Zhu.[1] As such, obtaining the list of associated candidates for each document was a trivial task. HTML tags were retained in the index for ease of implementation; we assume that any degradation of retrieval performance will have the same impact on all implemented expert finders.

4.2.2 Findwise dataset

The Findwise dataset used in this study is made up of documents from the Findwise internal network. Internal documents within the company are split up into different repositories. Out of these we used those repositories searchable from Findwise's central search system, which at the time of writing was under active development. The two repositories used were an organizational information repository and a developer wiki.

The organizational information repository contains information about competencies, company goals, methodologies, organization, offices and events. The wiki contains information for developers, such as information about internal projects, platforms or development tools. At the time of writing, the organizational info repository contained 321 documents, and the wiki contained 355. In this dataset, HTML tags were stripped out.

Findwise also has a repository of personal pages, implemented in an enterprise social network. These personal pages contain contact information, and allow employees to add skill keywords to their profiles, which are searchable. This system allowed us to obtain a list of all 98 employees of Findwise. Note that the skill information in this repository was not used as expertise evidence by the automatic expert finders studied.

Candidate-document association The lists of candidates associated with each document were created in slightly different ways depending on the repository. In both repositories, a simple named entity recognition step was used to locate candidates in the text before stripping out HTML tags. We took the approach of using regular expressions to match the email or full name of all employees, allowing up to five characters between the first name and last name of employees. This is similar

[1] For a description of how this was done, see [27]

to how this problem was solved by Zhu for the pre-tagged W3C corpus. In our case however, no attempt was made to match different name variations such as shortened first names, as Zhu did [27].

In the organizational info repository, some 78% of documents have author information, which was added to the list of associated candidates. Similarly, all wiki articles contain "last modified by" information which was also incorporated into the list of associated candidates.

Test topics and ground truth Test topics were obtained from self-reported skill keywords on employee personal pages. After stripping out all keywords which only occurred once, 29 topics was left. Roughly 60% of these topics were technical terms, while the others ranged from management or marketing skills to the names of particular clients of the company.

This sparse data did not enable distinction between missing skill information and actual lack of a particular skill. In order to complete the data and create a more realistic ground truth, all employees of the company was asked via email to rank their skills on the 29 different topics. On each topic, the employee was asked to rate his or her skill/knowledge level as either *I don't know this*, *novice*, *advanced* or *expert*. 36% of employees filled out the survey.

For measurements requiring binary relevance, only employees with skill level *expert* and *advanced* were classified as relevant, unless otherwise specified.

4.3 Measurements

We will judge the performance of the expert finder systems developed on four metrics based on binary relevance; MAP, P@5, P@10 and R-precision. Additionally, for the Findwise dataset where we enjoy the luxury of graded relevance judgments, we will also look at the nDCG. All of these metrics are common in research, and tend to give a good overview of performance [21].

For the Findwise dataset, candidates lacking expertise judgments for a topic was removed from the retrieved results prior to calculating scores. All measurements were calculated using the `TREC_EVAL` program.[2]

MAP *Mean Average Precision* is a single-figure measure of quality across recall levels. For a single information need, MAP is the average of the precision values after each relevant candidate has been retrieved. For multiple information needs, the MAP is simply the average of the MAP values of each individual information need.

[2] `trec_eval` is an IR evaluation program commonly used in research. It is available on the TREC web site `http://trec.nist.gov`.

Formally, if Q is the set of test queries, and each $q_j \in Q$ has an associated list of relevant candidates $\left\{c_1, ...c_{m_j}\right\}$, then

$$\text{MAP(Q)} = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{jk})$$

where R_{jk} is the set of ranked results from the top result to candidate c_k.

MAP has been shown to provide accurate measurements of the performance difference between different IR methods, even when relativity few test topics are available [8].

P@*k* MAP is a good measure for the retrieval quality over all retrieved results, but in some scenarios this might not be a good model of what a user wants. *Precision at* k, abbreviated P@k, is simply the precision after k candidates have been retrieved, where k is any positive integer. Thus, P@k models a user who will not look at more than k candidates. For multiple information needs, P@k is simply the arithmetic mean of all the individual P@k scores. [20, p. 161]

In this report, the scores for P@5 and P@10 are presented.

R-precision P@k does not always average well. The number of relevant candidates for an information need (call this $|rel|$) can differ significantly from one information need to another, significantly affecting the P@k measure. This is a problem for the Findwise dataset in particular, where the set rel is very small for some topics. For instance, if $|rel| = 3$, even a perfect expert finder will have a P@10 of only 0.3.

R-precision solves this by measuring the precision after $|rel|$ documents have been retrieved; thus making sure that a perfect system always gets a score of 1. This means that, for a single information need, R-precision is equivalent to P@$|rel|$. [20, p. 161]

nDCG Unlike the previously detailed metrics, *normalized discounted cumulative gain* takes into account graded relevance judgments. First, each relevance level gets a numeric value assigned to it. Then, we sum the relevance values of each retrieved candidate, but with a logarithmically increasing discount factor. In this way, nDCG rewards highly relevant documents retrieved early.

Assessment	R
I don't know this	0
Novice	1
Advanced	3
Expert	10

Table 4.3: Numeric values assigned to the relevance judgments for nDCG.

Formally, if $R(j, n)$ is the relevance score given to the candidate in the n:th place in the results for query q_j:

$$\text{nDCG}(Q, k) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Z_{kj} \sum_{n=1}^{k} \frac{2^{R(j,n)} - 1}{\log_2(1+n)}$$

where k is the number of retrieved candidates, and Z_{kj} is a normalization factor calculated to make it so that a perfect ranking's nDCG for query j is 1. [20, p.163]

The numeric values assigned to the relevance assessments are listed in table 4.3.

Chapter 5

Experimental results

Here, we will present results comparing the different expert finders implemented. We will also show a comparison between the best of the implemented expert finders and a system based on manually curated expertise information.

5.1 Comparison of automatic expert finders

5.1.1 W3C corpus

Tables 5.1 and 5.2 show performance metrics for the different expert finders implemented, and also includes the top TREC submission for the corresponding year.

Table 5.1: W3C corpus, 2005 test topics.

Finder	MAP		P@5		P@10		R-precision	
Profiles	0.143	−25%	0.192	−44%	0.236	−21%	0.192	−22%
Ranks	0.203	+7%	0.340	−1%	0.298	−1%	**0.261**	+6%
Votes	0.184	−3%	0.340	−1%	0.300	±0%	0.238	−4%
CombSUM	0.192	+1%	0.360	+5%	**0.322**	+7%	0.257	+4%
ExpSUM	0.187	−2%	0.344	±0%	0.298	−1%	0.238	−4%
ExpMNZ	0.187	−1%	0.344	±0%	0.300	±0%	0.237	−4%
NormExpSUM	**0.208**	+10%	**0.376**	+9%	0.312	+4%	0.256	+4%
NormExpMNZ	0.200	+6%	0.364	+6%	0.320	+7%	0.256	+4%
THUENT0505	0.275		0.488		0.452		0.333	

The percentages describe the difference from the median value of the implemented finders. Numbers in bold indicate the top score for each metric. The 2005 winner of the TREC enterprise track *THUENT0505* is not included in the median nor the selection of the top score. MAP values are plotted in figure 5.1.

Figure 5.1: W3C 2005, MAP scores.

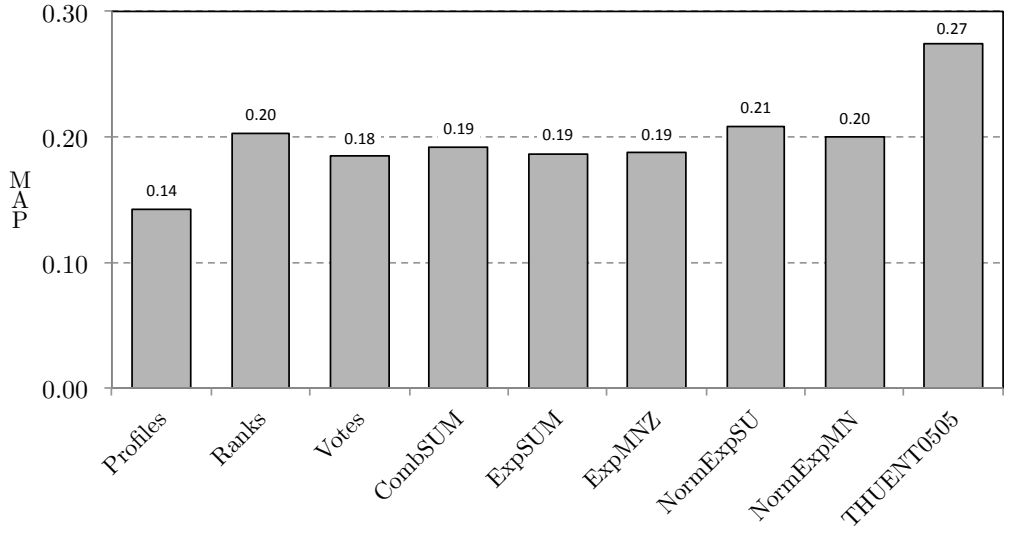


Table 5.2: W3C corpus, 2006 test topics

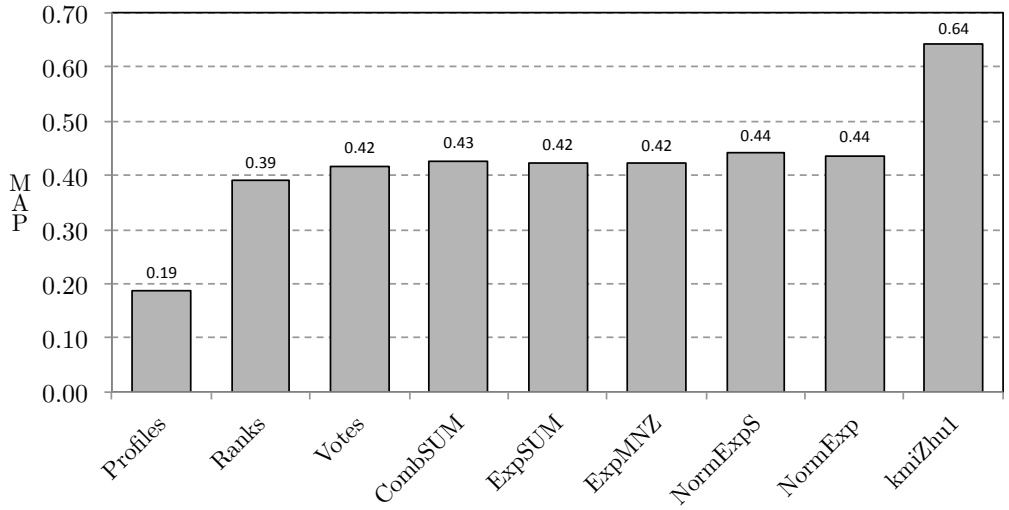
Finder	MAP		P@5		P@10		R-precision	
Profiles	0.188	−56%	0.371	−51%	0.394	−42%	0.261	−43%
Ranks	0.390	−7%	0.727	−5%	0.631	−7%	0.432	−5%
Votes	0.417	−1%	0.763	+0%	0.669	−2%	0.445	−2%
CombSUM	0.427	+1%	0.776	+2%	0.682	+0%	0.465	+3%
ExpSUM	0.423	+0%	0.759	−0%	0.680	±0%	0.454	+0%
ExpMNZ	0.421	−0%	0.759	−0%	0.680	±0%	0.453	−0%
NormExpSUM	**0.440**	+4%	**0.800**	+5%	**0.684**	+1%	0.464	+2%
NormExpMNZ	0.437	+3%	0.784	+3%	0.682	+0%	**0.473**	+4%
kmiZhu1	0.643		0.825		0.735		0.624	

The 2006 winner of the TREC enterprise track *kmiZhu1* is not included in the median nor the selection of the top score. MAP values are plotted in figure 5.2.

Comments In tables 5.1 and 5.2 we can clearly see that all of the document-based methods outperform the profile-based one on the W3C corpus. The performance difference between each individual document-based method and the profile-based one was shown to be statistically significant for all listed measures, using Student's t-test at significance level $p = 5\%$ for the 2005 test topics, and $p = 0.01\%$ for the 2006 test topics.

Among the document-based methods, the NormExpSUM finder seems to produce the best results overall, as it produces the best MAP on both the 2005 and

Figure 5.2: W3C 2006, MAP scores.



2006 test topics, while also producing consistently high scores for the other metrics. For the 2005 test topics, NormExpSUM was shown to offer statistically significant MAP improvements at $p = 10\%$ over the ExpMNZ, ExpSUM, scores and voted finders; for the 2006 test topics the improvement was statistically significant only for the ranks and votes methods.

5.1.2 Findwise corpus

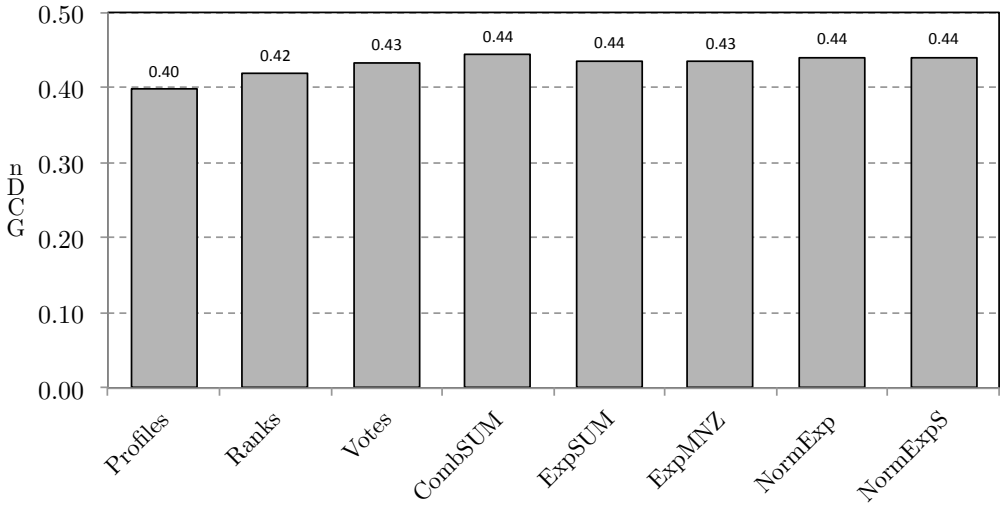
Table 5.3 shows the same metrics as tables 5.1 and 5.2, with the addition of nDCG. Figure 5.3 show a plot of the nDCG values.

Table 5.3: Findwise corpus

Finder	MAP		P@5		P@10		R-precision		nDCG	
Profiles	0.268	−7%	0.393	−4%	0.297	−1%	0.313	−5%	0.397	−9%
Ranks	0.271	−6%	0.428	+4%	0.307	+2%	0.331	+0%	0.418	−4%
Votes	0.285	−1%	0.400	−3%	**0.310**	+3%	0.328	−1%	0.433	−0%
CombSUM	**0.308**	+7%	0.428	+4%	0.300	±0%	0.358	+9%	**0.443**	+2%
ExpSUM	0.291	+1%	0.407	−1%	0.300	±0%	0.329	−0%	0.435	+0%
ExpMNZ	0.287	−1%	0.407	−1%	0.300	±0%	0.329	−0%	0.434	−0%
NormExpSUM	0.301	+4%	**0.441**	+8%	0.300	±0%	**0.361**	+9%	0.440	+1%
NormExpMNZ	0.297	+3%	0.414	+1%	0.303	+1%	0.335	+2%	0.441	+1%

Comments Table 5.3 shows that on the Findwise corpus, the gap in performance between the document and profile-based finders is narrowed, although the profile-based finder still performs worse across the board. On this dataset, it is not clear

Figure 5.3: Findwise corpus, nDCG scores.



which finder performs the best; scores, NormExpMNZ and NormExpSUM seems to be in the top, but the difference compared to ExpMNZ, ExpSUM and Voted is extremely small. None of the differences in relative performance for this dataset could be statistically verified at $p = 10\%$ for any metric.

5.2 Automated versus manually curated expert finders

Table 5.4 shows a comparison between the *NormExpSUM* automatic expert finder and the personal pages based searcher on the Findwise dataset. We list the recall, calculated over all non-zero scoring candidates, and the R-precision. Figure 5.4 shows the distribution of the R-precision scores per query.

Table 5.4: Automated expert finding versus personal pages

Definition of relevant	Recall			R-precision		
	Personal profiles	NormExp-SUM	diff	Personal profiles	NormExp-SUM	diff
expert	0.200	0.423	+11%	0.200*	0.235	+17%
& advanced	0.134	0.414	+210%	0.134*	0.361	+170%
& novice	0.063	0.401	+542%	0.063*	0.385	+516%

* The fact that the recall and r-precision values are identical for the personal pages is not an error or typo. This is caused by the fact that when only relevant candidates are returned, as is the case here, the formulas for calculating recall and r-precision are equivalent.

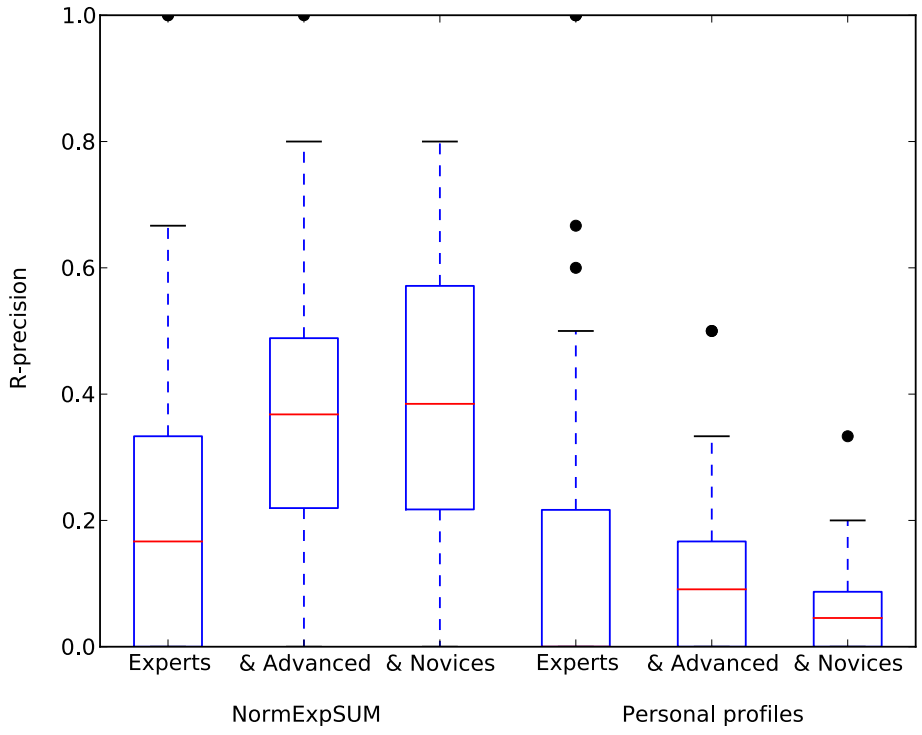
Comments In table 5.4 it is plain to see that our automatic expert finder vastly outperforms the personal pages-based expert search system in terms of recall. The difference becomes even more pronounced when we expand our definition of relevant from only the self-described experts to the advanced and novice skill levels.

However, the personal pages-based system does have the advantage that all candidates returned by it are sure to be relevant; after all, it is based on self-reported skills which we used the ground truth. In practice we would most likely opt to give the end user a combination of the results from both, should we have both these systems in place. Obviously, we would return the personal pages-based results first, as they are almost certainly relevant.

We should also make a note that the test topics used in calculating these measures were obtained from the same data that the personal pages searcher bases its results on. Given any test topics not explicitly entered by the employees, the personal pages-based searcher would obviously fail to return any results at all. On the other hand, our automatic expert finder can be expected to generalize to a wide spectrum of possible queries.

Score variance Figure 5.4 show that for both the automatic and manual expert finder the per-query score variance is quite large. For many queries, the precision was either zero or one.

Figure 5.4: Distributions of R-precision scores. The boxes show the quartiles, and the whiskers extend to the most extreme data points in the 1.5 inner quartile range, i.e. the central 75% of data points. The dots represent outliers.



Chapter 6

Discussion

In this section, the experimental results are interpreted, and put in a practical context.

6.1 Best automatic expert finder

Retrieval performance Out of the expert finders examined in this study, it is clear that the document-based methods work better than the profile-based one. This is most likely caused by the fact that when judging each candidate's relevance to a query, all text related to the candidate is equally weighted, whereas the document-based methods only consider text related to the query. Intuitively, the profile-based finder's document merging step causes an information loss, as it "forgets" that each document is about a certain topic.

If this is correct, we should observe that the profile-based method's main drawback is a loss of the often crucial early precision. Furthermore, this gap in precision should become narrower as more candidates are retrieved. Also, we would expect the profile-based finder's disadvantage to be less pronounced for the Findwise dataset, where there are fewer documents that should lessen the information loss in the profile creation step. This is in fact what the experimental results show, providing a reasonable explanation for the document-based methods' superior performance.

Ease of implementation The document-based methods have a distinct advantage over profile-based algorithm in terms of difficulty of implementation. Profile-based methods require us to create a brand new index of candidate profiles; this stands in stark contrast to the document-based methods, which can easily be deployed on top of an existing document retrieval system by reusing its index.

Furthermore, improvements to the retrieval quality of the document search system will directly benefit a document-based expert finder along with the document search. On the other hand, improving the query matching of a profile-based system is a separate problem.

An additional advantage of document-based methods is that producing a list of supporting documents becomes a trivial task. Doing this can increase the transparency of the system and provide analysis support for the expert seeker.

Best expert finder overall Among the different document-based methods, it would appear that the NormExpSUM method is preferable to the other approaches. On the Findwise dataset however, it performed no better than, for example, the simpler CombSUM analyzer. This suggests that the choice of analyzer becomes less crucial for small corpora. However, even though the score assignment algorithm for NormExpSUM is slightly more complex and thus harder to understand than, for instance, CombSUM, the additional work needed to implement NormExpSUM is very small, bordering on trivial. As such, it might make sense to use NormExpSUM even in small datasets, as this equips the system to handle any future growth of the corpus in an optimal manner.

6.2 Practical usability

The practical usefulness of the system to a real world user cannot be determined exactly, as it contains many subjective and hard to measure qualities. This is especially true when feedback from real users is not available, as was the case in this study. However, we can reach an estimate by considering the value of correct and incorrect results, along with the precision and recall of the system.

As detailed in chapter 2, easily finding an expert can be of high importance. Thus, the reward for correctly finding an expert, i.e. a true positive, can be very large. Since the use case of locating candidates *without* a certain competence is rare, true negatives have a small reward.

When it comes to false positives, the associated cost is usually low; the seeker can simply ask the suggested expert if she has the desired expertise, and receive a negative answer. In fact, the candidate might be able to point the seeker to an actual expert; because she appears in documents related to the query, it is also likely that she is organizationally close to an actual expert.

Furthermore, if the expert search system has sufficiently good analysis support, the seeker can exclude false positives before contacting them, decreasing the cost even further.

False negatives are usually not a big problem either, since in the most common use cases only one expert is needed. One obvious exception is when the seeker is attempting to put together a large team of similar experts.

Taking into account the rewards and costs above, the scores presented in chapter 5 are sufficiently high for us to claim, with confidence, that the expert search systems developed are practically useful to users in organizations where locating expertise is an important task.

6.3 Comparison with state of the art approaches

As we saw in tables 5.1 and 5.2, the top scoring submissions to the TREC conference for both 2005 and 2006 vastly outperform the approaches taken in this report. However, we should note that both of these top-scoring approaches take advantage of various document and collection-specific heuristics, as described in section 3.3. Such approaches as tuning the system to specifically identify members of working groups, as done by Fu et al., will clearly not generalize to a wider spectrum of queries or corpora. An organization might not even have working groups, and queries for members of working groups make up only a fraction of all possible queries.

Also, many of the collection specific heuristics are based on the web crawl nature of the data, containing no author information. One example of such heuristics are candidate-query proximity based methods, used by both Fu at al. and Zhu et.al. While this approach has been shown to greatly improve performance, it is not obvious how to generalize such an approach to a corpus with characteristics similar to the Findwise dataset, where a large portion of the document-candidate associations come not from the document text itself but from author tags.

With this in mind, we can feel slightly better about the comparative performance of the expert finders implemented for this study. They all use a minimal number of domain-specific heuristics, and can thus be expected to generalize well to new datasets and queries. Still, we are forced to conclude that additional improvements to the algorithms implemented here would be necessary in order to compete with state of the art algorithms. A few such potential improvements are discussed in chapter 8.

6.4 Automated versus manual expert finding systems

As was shown in section 5.2, automatic expert finders offer expert retrieval results that are, at worst, comparable to the personal pages-based approach offered. For many, if not most queries, the performance of the automated system will be vastly superior to the manual approach. Such queries might include searching for experts on a specific project, persons with knowledge of a specific detail of a system, or very detailed expertise queries; in short, any topic that is mentioned in documents, but not likely to be explicitly entered into a manual system.

Dataset requirements for automated expert finding The automatic expert finders studied seem to work well even for small datasets, such as the Findwise dataset used here, which contained only ~600 documents. However, the quality of the documents in the dataset with regard to how well they indicate expertise is likely at least as important as the sheer number of documents. As we observed in section 5.2, some of the test queries perform very well, and some extremely poorly. The negative extremes are most likely due to the small size of the corpus; there might be very few documents pertaining to a particular query, and these might not

contain mentions of any experts. On the positive end of the spectrum, the system most likely located a specific high-quality page about a certain competence area group and its members. The Findwise dataset contains many such high quality documents, such as pages concerning specific teams or competencies, which most likely compensate for the small corpus size.

In general we expect high quality documents to contain information that might be relevant to a specific expertise query, while also allowing candidates to be connected with it. Examples of such high quality documents might be email conversations, resumes, or project specifications. However, this is purely speculative, as we have not conducted any experiments directed at discerning which types of documents contain the most valuable expertise evidence.

Analysis support One problem that the automatic expert finders studied do not address, which personal pages do, is creating human readable expert profiles. It is important that the user of a system gets analysis support to figure out which suggested experts are the most relevant. The document-based expert finders studied can provide a list of supporting documents, which is a step on the way. However, it is also greatly beneficial to the user experience if the seeker is also presented with short descriptions of the experts, a feature which the automatic expert finders studied cannot provide on their own. Thus, despite the advantages of the automatic expert finders studied, they can not fully replace all the functionality of manually created personal pages. In most cases, the two systems would be best used together, with the automatic expert finder doing the bulk of the expert finding, and the personal pages supplying the expert profiles.

Chapter 7

Conclusions

We find that among the automatic expert finding systems developed, document-based methods work best. Among these, the NormExpSUM method offers stable and high performing results, thus being the preferred method. All document-based methods studied incur only a very small implementation overhead if a document search system is already in place, which is a reasonable assumption in practice.

Automated expert finding systems are an efficient and effective way to address the problem of locating experts, in many cases offering much better results than a manually curated system can. Furthermore, a manually curated system requires more work the more individuals are included, whereas an automatic system scales effortlessly to an increasing numbers of candidates.

However, automatic expert finders do not in general create human-readable expert profiles. As such they do not completely eliminate the need for manual systems such as personal pages or expert databases. Even so, automatic expert finders have a small implementation overhead and a high usability in practice. This allows them to be used as a cheap and efficient alternative to manual systems in many, if not all, regards. They can also provide a valuable complement to manual systems, by providing significant increases in recall.

Chapter 8

Future work

As we concluded in section 6.3, the expert finders implemented for this study have room for improvement. Here, some such potential improvements are listed. Some of these approaches have already been experimentally evaluated in research. However, most of the evaluation in research was done on the TREC enterprise datasets or datasets of similarly large sizes; it would be interesting to see if the results are reproducible on smaller corpora such as the Findwise dataset.

Also listed are some interesting related topics suitable for further research.

Other entity types The keen reader might have noticed that apart from the named entity recognition step where candidates were located in documents, no parts of the studied algorithms require the entities that are sought after to be persons. Thus, as long as a named entity recognition step can be designed for a particular type of entity, it can be searched for. It would be an interesting topic of further research to see how well the developed systems work with other types of entities such as places, teams or companies.

Expert profiling Expert profiling, that is, creating human-readable expert profiles from document data, might be automated. If such a system could archive sufficiently good results, it could be deployed in tandem with an expert finder and completely eliminate the need for a manually curated expertise system. Alternatively, the expert profiler might provide skill suggestions to the candidates, who could then more easily maintain their personal pages. An effective expert profiler could also be used to improve the results of the expert finder. Therefore, developing such a system would be a very attractive prospect.

Some research on the topic exists,[1] although much less than on expert finding. One of the reasons for this is likely the difficulty in objectively measuring performance.

[1] For instance Balog et al.[5] or Adomavicius et.al[1].

Data quality In section 6.4 we speculated in what kinds of data or documents are the most useful to an expert finder system. Since we expect some documents to contain better indicators of expertise than others, it would be interesting to perform experiments to discover just which types of data this is. The results might help us increase performance by weighting different types of documents differently.

Furthermore, in a practical setting all the available data on an internal network is usually not indexed and searchable. Here, the result of such an experiment would suggest what content is the most important to index from an expert finding perspective. For instance, say emails prove to be high quality expertise indicators. In that case, we might want to index email on a company mail server, even though this would likely not be useful in a document search scenario.

Improved query-document matching For this study, the query-document matching was done using a TF-IDF-based similarity measure. Other similarity measures, such as Okapi BM25 or language modeling approaches have been shown to have better retrieval performance in some cases [20, p.234][3]. Therefore, it would be interesting to see if the expert finding performance could be improved by changing the query-document matching algorithm, and if the same expert finding methods that worked best with the TF-IDF model remain the best in these cases.

Candidate priors We suspect that not all candidates are equally likely to be considered experts. It is possible to take advantage of this by introducing candidate priors. An example of this is the work of Shen et al., described in section 3.3. It would be interesting to see how their candidate PageRank algorithm generalize to a smaller dataset such as the Findwise corpus used in this study.

Document-candidate association models In this study, the document - candidate association model was a simple binary model, which assumed candidates either did or did not figure in documents. Improving this model to be more nuanced would likely lead to improvements in the overall performance of the system. There are a number of techniques that could accomplish this, for instance:

- *More nuanced candidate-document strength models*
  One potential way to improve the quality of the document candidate associations would be to take into account the number of times a candidate occurs in a document, and weigh candidates with many occurrences heavier. Also, we might weigh the candidates differently depending on how they are associated with the document; for instance, if the author of a document is known, this author could be given more weight than candidates mentioned inside the document. Similarly, a candidate whose full name was found in the document might get more weight than a candidate for whom only the email address was found.

- *Proximity-based methods*
  Methods based on candidate-query proximity such as window-based methods are commonly used in literature, and have been shown to improve results significantly in some cases, particularly when using profile-based methods. [3][17][28][24][6]

- *Document-profile similarities*
  An even more advanced approach usable in document-based methods is to calculate a relatedness measure between documents and candidate profiles. This was done in the TREC 2008 top-scoring algorithm by Balog et.al, described in section 3.3.

Personalization and related experts Sometimes, it might be of interest for the expert seeker to find candidates organizationally or socially close to each other, for instance to assemble a team. If the seeker is already a candidate in the system, the same closeness measures could be employed to personalize his or her results.

To accomplish this, one approach might be to cluster experts with similar profiles together. Another would be to analyze the graph formed by co-occurrences of candidates in documents, or the graph formed by email communications, as done by Schwartz and Wood [22].

Bibliography

[1] G. Adomavicius and A. Tuzhilin. "User profiling in personalization applications through rule discovery and validation". In: *Conference on Knowledge Discovery in Data: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. Vol. 15. 1999, pp. 377–381.

[2] J. A. Aslam and M. Montague. "Models for metasearch". In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '01. New Orleans, Louisiana, USA: ACM, 2001, pp. 276–284. ISBN: 1-58113-331-6. DOI: 10.1145/383952.384007. URL: http://doi.acm.org/10.1145/383952.384007.

[3] K. Balog, L. Azzopardi, and M. de Rijke. "A language modeling framework for expert finding". In: *Information Processing & Management* 45.1 (2009), pp. 1–19.

[4] K. Balog and M. De Rijke. *Combining candidate and document models for expert search*. Tech. rep. DTIC Document, 2008.

[5] K. Balog and M. De Rijke. "Determining expert profiles (with an application to expert finding)". In: *Proceedings of the 20th international joint conference on Artifical intelligence*. Morgan Kaufmann Publishers Inc. 2007, pp. 2657–2662.

[6] K. Balog et al. *Overview of the TREC 2008 enterprise track*. Tech. rep. DTIC Document, 2008.

[7] N. J. Belkin et al. "Combining the evidence of multiple query representations for information retrieval". In: *Information Processing & Management* 31.3 (1995), pp. 431–448.

[8] C. Buckley and E. M. Voorhees. "Evaluating evaluation measure stability". In: *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '00. Athens, Greece: ACM, 2000, pp. 33–40. ISBN: 1-58113-226-3. DOI: 10.1145/345508.345543. URL: http://doi.acm.org/10.1145/345508.345543.

[9] N. Craswell, A. de Vries, and I. Soboroff. "Overview of the trec-2005 enterprise track". In: *TREC 2005 conference notebook*. 2005, pp. 199–205.

[10] A. Forward and T. Lethbridge. "The relevance of software documentation, tools and technologies: a survey". In: *Proceedings of the 2002 ACM symposium on Document engineering*. ACM. 2002, pp. 26–33.

[11] Y. Fu et al. "Thuir at trec 2005: Enterprise track". In: *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005)*. 2005.

[12] I. Greif. "Everyone is Talking About Knowledge Management (Panel)". In: *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*. CSCW '98. Seattle, Washington, USA: ACM, 1998, pp. 405–406. ISBN: 1-58113-009-0. DOI: 10.1145/289444.289516. URL: http://doi.acm.org/10.1145/289444.289516.

[13] D. Hawking. "Challenges in enterprise search". In: *Proceedings of the 15th Australasian database conference - Volume 27*. ADC '04. Dunedin, New Zealand: Australian Computer Society, Inc., 2004, pp. 15–24. URL: http://dl.acm.org/citation.cfm?id=1012294.1012297.

[14] B. He et al. *University of glasgow at trec 2008: Experiments in blog, enterprise, and relevance feedback tracks with terrier*. Tech. rep. DTIC Document, 2008.

[15] H. Kautz, B. Selman, A. Milewski, et al. "Agent amplified communication". In: *Proceedings of the National Conference on Artificial Intelligence*. 1996, pp. 3–9.

[16] C. Kuhlthau. "A principle of uncertainty for information seeking". In: *Journal of Documentation* 49.4 (1993), pp. 339–355.

[17] W. Lu et al. "Window-based enterprise expert search". In: *Proceeddings of the 15th Text REtrieval Conference (TREC 2006)*. 2006.

[18] *Lucene API documentation*. URL: http://lucene.apache.org/core/3_6_0/api/all/org/apache/lucene/search/Similarity.html (visited on 05/29/2013).

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

[20] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. ISBN: 978-0-521-86571-5.

[21] F. Radlinski and N. Craswell. "Comparing the sensitivity of information retrieval metrics". In: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2010, pp. 667–674.

[22] M. Schwartz and D. Wood. "Discovering shared interests using graph analysis". In: *Communications of the ACM* 36.8 (1993), pp. 78–89.

[23] H. Shen et al. *Research on enterprise track of TREC 2008*. Tech. rep. DTIC Document, 2008.

[24] I. Soboroff, A. de Vries, and N. Craswell. "Overview of the trec 2006 enterprise track". In: *TREC 2006 Working Notes* (2006).

[25] A. Vivacqua. "Agents for expertise location". In: *Proc. 1999 AAAI Spring Symposium Workshop on Intelligent Agents in Cyberspace*. 1999, pp. 9–13.

[26] D. Yimam-Seid and A. Kobsa. "Expert-finding systems for organizations: Problem and domain analysis and the DEMOIR approach". In: *Journal of Organizational Computing and Electronic Commerce* 13.1 (2003), pp. 1–24.

[27] J. Zhu. *W3C Corpus Annotated with W3C People Identity*. Sept. 2006. URL: `http://ir.nist.gov/w3c/contrib/W3Ctagged.html`.

[28] J. Zhu et al. "The Open University at TREC 2006 Enterprise Track Expert Search Task." In: *TREC*. Ed. by E. M. Voorhees and L. P. Buckland. Vol. Special Publication 500-272. National Institute of Standards and Technology (NIST), 2006. URL: `http://dblp.uni-trier.de/db/conf/trec/trec2006.html#ZhuSREM06`.