THUIR at TREC 2005: Enterprise Track[1]

Yupeng Fu, Wei Yu, Yize Li, Yiqun Liu, Min Zhang, Shaoping Ma
State Key Lab of Intelligent Tech. & Sys., CST Dept, Tsinghua University, Beijing 100084, China
Liuyiqun03@mails.tsinghua.edu.cn

Abstract: IR group of Tsinghua University participated in the expert finding task of TREC2005 enterprise track this year. We developed a novel method which is called document reorganization to solve the problem of locating expert for certain query topics. This method collects and combines related information from different media formats to organize a document which describes an expert candidate. This method proves both effective and efficient for expert finding task. Our submitted run (THUENT0505) obtains the best performance in all participants with evaluation metric MAP. The reorganized documents are also significantly smaller in size than the original corpus.

1 Introduction

Expert Finding is a new sub task in TREC Enterprise Track. It simulates a task that often appears in practical enterprise search applications. This Task is to find a person/group that satisfies a specific need in a multi-source database. Additionally, this person/group exists in a known list. For example, find someone in the list of employee names, who is familiar with Middle East businesses.

TREC 2005 Expert Finding Task is to help users find an expert in a certain topic of W3C organization. After user inputs a query of the topic, Expert Finding System will output a sorted name list according to persons' relevance to this query. The W3C corpus we use takes on a typical type of enterprise data, which is, not huge-scaled, with data of various kinds like Email, WEB Page and Wiki. The candidate list includes 1092 W3C experts' names. Every item of the list has an exclusive person-id, just like the name list of all employees in corporation.

A possible solution for this problem will be: Retrieve documents in the given corpus using query and then rank experts according to their appearances in result documents. This voting-based method is easy to carry out but may not fit for expert finding very much. First, top ranked documents retrieved with a certain query topic are likely to be documents which describe this topic in detail. These documents may not contain information about experts in this topic. For example, an article on an XML expert will be more useful for the query "XML" in expert finding instead of an article describing XML schema and related technologies; but the latter one is more likely to be given a higher retrieval status value using traditional IR technologies. Second, it is difficult for this method to deal with multi-source information. We are lack of reliable techniques to combine retrieval results from different types of documents [1][2].

In this paper we propose a new method to solve expert finding problem. The method is

[1] This work is supported by the Chinese National Key Foundation Research & Development Plan (2004CB318108), Natural Science Foundation (60223004, 60321002, 60303005, 60503064) and the Key Project of Chinese Ministry of Education (No. 104236)

called “document reorganization” because it reorganizes description from all sources of information for each candidate expert. It is performed as follows:

(1) Mark up appearance of all candidates in the given corpus;
(2) Extract descriptions for candidates using different rules according to their appearances in the corpus and the document type;
(3) Combine these descriptions and then get a combination of descriptions for each expert. Each newly-formed combination (also a “document”) is corresponding to an entity.

The key idea of this reorganization method is to assemble information. We cleanse data from different document types, and assemble information related to the candidate. This method reduces the complexity of search space and has a strong scalability for different document types.

2 Candidate Identity Extraction and Tagging

It is important to mark up the candidate in different data sources according to the given name list. We must clarify some definitions as follows:

Candidate identity: a candidate may have one or more identities. An identity could be his email, nick, cell phone number and so on. If this identity appears in a document or somewhere, we can recognize the appearance as this candidate.

Candidate name-email relationship: a candidate may have different names and emails. While representing them, we usually use a name-email pair. This may be useful for identifying people because if we have pairs (A,B) and (B,C), we will know that (A,C) is also possible. We use this property to educe all identities of a candidate.

2.1 Candidate Identity Extraction

Starting with the 1092 candidates’ name-email relationships, we go through all the “from”, “to”, “cc” fields in the processed mail lists. Each field will contain one candidate name-email relationship. If the name and the email in the relationship are different identities of two people, then merge all the identities from the two people into a new set and make it a new candidate. With this method, we got an identity set which was made up of variants names of 1092 candidates given.

This process is sometimes difficult because of data quality problem in the corpus. For instance, some one may borrow others’ Email address, or some one spell the wrong name, or some ones share the same abbreviates for their name. In order to solve these problems, several heuristics are used:

(1) Candidate identity with less than 3 letters is highly duplicated, and it is unreliable.
(2) Candidate name-email relationship with less than 3 appearances is likely to be a typo, and it is unreliable.
(3) Some frequently-used names such as Tom or David have a larger opportunity to be a name in common, and it is unreliable.

After that, we will get the candidate identities. The result will be all possible identities for

the 1092 candidates in this corpus.

2.2 Candidate Identity Tagging

First we build a DFA according to the Aho-Corasick by using all the candidate identities extracted. Then, we run the DFA on the document to process. On the positions where the DFA accepts, we do backtrack to find the start position of the accepted identity. At last we replace the identity with a marked identity such like <candidate id="candidate-0001">dan brickley</candidate>) for future use.

Elder tools such like regular expressions just can not work here because they are not optimized for matching a large set of identities in reasonable time. So we chose Aho-Corasick algorithm for this task.

Aho-Corasick algorithm [3] is an algorithm for matching a set of keywords in text. It is similar to the KMP algorithm, and run in O(m+n) time in worst case. Where n is the length of the text and m is the total length of all the keywords. The main principle of the algorithm is to build a DFA and run the DFA on the text to obtain the result. However, our task requires more than keyword matching, because both "John-smith" and "john smith" are the same as "john smith". So our algorithm must handle spaces carefully.

For the same reason, backtrack is required for the algorithm, because we do not know that where "john smith" ends, whether the end of "John-smith" or the end of "john smith". So we must do the backtrack to find the beginning of the identity.

At last, candidate identities may overlap each other. Our strategy is to accept the one ends latest but throw the others.

3 Document Reorganization

Reorganization of the description documents means reorganizing documents with information from different sources. We extract information that may be description among the candidate identity in documents and combine it to form one document. Because of the variety of data sources, the methods of information extraction are different from each other. At the same time, it is also necessary to find potential description by the characteristic of documents. In this experiment, data sources are the W3C corpus and its composition is as follows (according to [4]):

Type	Scope	Size (GB)	Docs	avdocsize (KB)
Email	lists	1.855	198,394	9.8
Code	dev	2.578	62,509	43.2
Web	www	1.043	45,975	23.8
Wiki web	esw	0.181	19,605	9.7
Misc	*other*	0.047	3,538	14.1
Web	people	0.003	1,016	3.6
	all	5.7	331,037	18.1

Figure 1 Data Types in W3C corpus

We can see that there are mainly three types of documents in this corpus: Email, Web pages (Web, Wiki web and Misc) and text of codes. We processed Email and Web page in this task because little information on experts can be obtained from coding text.

3.1 Email Processing

The purpose for this step is to get rid of redundant information and extract information on candidates in E-mails for future use.

We take all the pages from "lists-*" in w3c corpus, and perform a three-step process:

(1) Parse the document into a valid TREC Web format document.

(2) Take out the body field from the TREC Web format document, then pass it into several filters in order to extract certain fields such like the "from" field, "subject" field, and "date" field.

(3) The body text of the email is extracted and all the HTML tags in it are stripped.

The final result will be a combination of step 2 & 3.

The difficulties lie in the format of the mails. The mails are not stored according to the RFC. They are generated HTML files from the mail archives. The following points are important.

First, the encoding of the text varies. This might not be a serious problem when processing English documents. However, several candidates have non-English names. So if we cannot handle the encoding correctly, we may miss these candidates. Fortunately, all the characters in candidates' identities are in ISO-8859-1. So we can simply parse all the documents in ISO-8859-1.

In supplemental for the above point, HTML encodings also matters. "é", for example, is an HTML entity for character "é". They should be the same for the program. So HTML decoding must be done before the next step.

Second, that's quite a complex problem to extract mail fields from the HTML page. The fields include mail's subject, date, from, to, cc, mail body, etc. Generated HTML pages store these fields in a variety of formats. They are similar in general, but various in details.

Our way of handling this problem is to pass the text through a set of filters. Each filter will handle only one case. And each filter may have some sub processors which just handle one or two fields in the filter's case. This way will make the problem easy to strike by using the case by case method.

At last, several trivial points must be cleared. We must mark up each mail's position in a thread, this is done by using the document tree ID file generated by Bob Allen (http://cio.nist.gov/esd/emaildir/lists/trec-ent/msg00067.html). And we should strip replies from mail bodies, I mean the lines started with ">", in order to reduce redundant.

Additionally, we also pick Metadata and paragraphs in which candidate identity appears.

3.2 Web Page Processing

We extract information from documents in which candidate identities were marked. For each occurrence of the candidate in the page, we use the following sources to form a document as

candidate’s description.

(1) The title of the document.

(2) The head line 1, head line 2, bold character line nearest to the occurrence position;

(3) Window information: The contents around the occurrence position of the candidate name;

(4) Group information, specific treatment of some useful web page.

We merged Web page information ,mail information ,in-link anchor text together as description document.

The difficulty lies in how to set range of the window. Too wide window may contain useless even wrong information, and on the other side, narrow window may lose information.

At first we tried to set the range to a default value; however, the result is coarse and not reliable. We tried several values over the training topics, and at last we retrieve 10 words ahead and 10 words after the position as window information.

Some Web pages in the corpus we found the phenomena that names always appear as groups. Some groups don't have description for individual while other groups have. So we found the groups information and retrieve the group information. In a W3C corpus, groups information frequently found and is great helpful. However, the accuracy is based on whether all the name entity is identified. If we ignore or forget to mark some name entity, the group may be parted, and the group information is lost and misleading.

We changed the data structure and greatly reduce the indexing amount. The group information improved the result to a high level.

3.3 Information Extraction for Special Web Pages

Some special pages may contain more information than other pages. We do special treatments for them. The page we processed actually is http://www.w3.org/People/all. Structurally analysis is done to this page, and information about the people are all extracted, such like their positions and instructions.

4 Experiment Results

Table 4.1 contains the result of comparative experiment. We use a voting-based method, which directly finds candidate identity from the first *n* documents returned, and then ranks candidate by the weight of returned documents. The best result is obtained when *n* is set to 100.

Table 4.1 MAP of traditional search method

N	MAP
100	0.1694
200	0.1592
150	0.1643
120	0.1728
80	0.1671
50	0.1626

Table 4.2 contains information of description documents got by information extracting from all data sources. In the table, “Size” expresses the size of former document; “DesDocSize” expresses the size of description document. DesDocSize is about 4% of the original corpus size.

Table 4.2 Information of description documents

SOURCE	Size	Docs	DesDoc	DesDocSize
WEB	1.043G	45975	872	43M
EMAIL	1.855G	198394	907	56M
PEOPLE	0.003G	1016	58	66K
Wiki	0.181G	19605	158	3.5M
Other	0.047	3538	558	1.7M

Table 4.3 contains results of searching in description documents. The five runs submitted are retrieved by our TMiner System, including:

Run1: This run makes use of all w3c web part information and inlink anchor text of these files. Text content are reconstructed and formed description files for each candidate person. Structure information inside web pages was also used to improve performance

Run2: This run makes use of all w3c web part information and inlink anchor text of these files. Text content are reconstructed and formed description files for each candidate person. Structure information inside web pages was also used to improve performance. Words from important pages are emphasized in this run.

Run3: This run makes use of all w3c web part information and inlink anchor text of these files. Text content are reconstructed and formed description files for each candidate person. Structure information inside web pages was also used to improve performance. Words from important pages are emphasized in this run. Word pairs are also given a higher weight.

Run4: This run makes use of all w3c web part information and Email lists (the list part) together with inlink anchor text of these files. Text content are reconstructed and formed description files for each candidate person. Structure information inside web pages was also used to improve performance. Words from important pages are emphasized in this run.

Run5: This run makes use of all w3c web part information and Email lists (the list part) together with inlink anchor text of these files. Text content are reconstructed and formed description files for each candidate person. Structure information inside web pages was also used to improve performance. Words from important pages are emphasized in this run. Bi-gram retrieval was also applied.

Table 4.3 Retrieval experiments based on document reorganization

Runs	MAP	Bpref	P@10
1	0.230	0.498	0.362
2	0.245	0.483	0.414
3	0.275	0.488	0.452
4	0.241	0.298	0.412
5	0.275	0.488	0.452

We can find from Table 4.3 and Table 4.1 that the document reorganization method gained much better performance than the voting-based method. The reorganization process can reduce unimportant information and combine useful description from different sources.

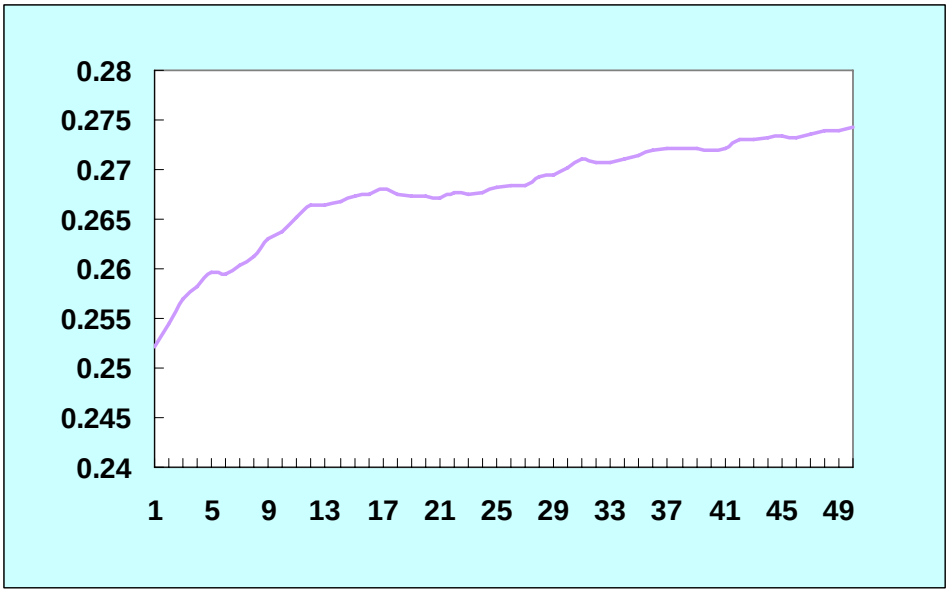
We also tried bi-gram retrieval in our experiments. Table 4.4 and Figure 4.1 show the

change of MAPs got by bi-gram retrieval on reorganized documents. *Lamda* is the weight of retrieval status value using bi-gram retrieval. The experiment results show that MAP will be 12.5% higher if we use bi-gram weighting and set *Lamda* to a proper value. It is different from retrieval experiments for traditional web search tasks such as web tracks in TREC2003 [5] and TREC2004 [6], in which bi-gram weighting gives ranking little affection. It may be explained by the fact that the reorganized documents are much smaller than Web documents and bi-gram weighting can significantly improves retrieval precision for short documents.

Table 4.4 Bi-gram retrieval on reorganized documents

Lamda	MAP	Lamda	MAP
0	0.245	25	0.2683
5	0.2597	30	0.2702
10	0.2638	35	0.2714
15	0.2674	40	0.272
20	0.2673	45	0.2734

Figure 4.1 Bi-gram retrieval on reorganized documents



5 Conclusions

Major conclusions of this paper can be concluded as follows:

(1) Document reorganization proves effective for expert finding task. It performs better than those traditional methods according to our experiment results. Our run in TREC2005 based on this method also gained the best performance in Expert Finding Task.

(2) Ranking reorganized documents by given different sources of context respective weights gets better performance than treating all sources the same. It accords our previous experiments in TREC web track [5][6] and is a possible solution for multi-source retrieval result combination.

(3) Bi-gram retrieval method does well by increasing precision for expert search, because

the size of reorganized documents is small and the query of common expert finding is professional and strongly detailed.

(4) In the process of documents reorganization, lots of unimportant and redundancy information is thrown off. The scale of new document set is far smaller than the former corpus. This can be regarded as a kind of data cleansing and is useful for practical enterprise information management.

In the future, we will try to apply this method to item-finding tasks in specific fields such as software search, MP3 search and so on.

References

[1] Broder, A.Z. and A.C. Ciccolo, Towards the next generation of enterprise search technology. IBM Systems Journal, 2004. **43**(3).

[2] David, H., Challenges in enterprise search, in Proceedings of the fifteenth conference on Australasian database - Volume 27. 2004, Australian Computer Society, Inc.: Dunedin, New Zealand.

[3] Aho, A. V. and Corasick, M. J. 1975. Efficient string matching: an aid to bibliographic search. Commun. ACM 18, 6 (Jun. 1975), 333-340.

[4] TREC enterprise track web site: http://www.ins.cwi.nl/projects/trec-ent/

[5] Min. Zhang, Chuan. Lin, Yiqun. Liu, Le. Zhao, Shaoping. Ma, THUIR at TREC 2003: Novelty, Robust and Web, in NIST Special Publication 500-255: The Twelfth Text REtrieval Conference (TREC 2003).

[6] Yiqun Liu, Canhui Wang, Min Zhang, Shaoping Ma, Finding "Abstract Fields" of Web Pages and Query Specific Retrieval--THUIR at TREC 2004 Web Track, NIST Special Publication: SP 500-261, Proceedings of the Thirteenth Text Retrieval Conference (TREC 2004).