Research on Enterprise Track of TREC 2007 at SJTU APEX Lab

Huizhong Duan[1], Qi Zhou[2], Zhen Lu[3], Ou Jin[4], Shenghua Bao[5], Yunbo Cao[6] and Yong Yu[7]

Apex Knowledge & Data Management Lab
308 Yifu Building, 800 Dongchuan Road, Shanghai, P.R. China
{ summer[1], jackson[2], luzhen[3], kingohm[4], shhbao[5], yyu[7]}@apex.sjtu.edu.cn,

Microsoft Research Asia
No.49 Zhichun Road, Beijing, P.R. China
{yunbo.cao[6]}@microsoft.com

1. Introduction.

This year we (APEX Lab, Shanghai Jiao Tong University) participated in both Document Search Task and Expert Search Task in Enterprise Track of TREC 2007.

For Document Search Task, we generally applied BM25 formula separately on different fields of HTML pages: Title, Anchor, H1, H2, Keywords, and Extracted Body. Various Static Ranking methods are also exploited. Scores are combined together using linear combination. Among all the techniques we have embedded in our system, our highlight is the static ranking approaches. Beside this, some data preprocessing methods and similarity function will also be introduced.

1. Static Ranking Approaches. Page quality is our focus for the task. Thus we studied various static ranking methods in Enterprise Corpus. Among them, PageRank[6] and Topic Sensitive PageRank[1], which both generate similar ranks for most pages, do not work for this Corpus. Then we research on HostRank[5]. The central problem of using HostRank is to define a host. After realizing that sub-portals of an enterprise do not necessarily earn a difference, we finally used sub-layers of sub-portals (AAA.BBB.CCC/DDD) as hosts.

2. Data preprocessing.

- **Title Extraction.** The title of a web document is a strong indicator of its topic, but only a small portion of pages are provided with such a manually written title. To conquer the problem we adopted an automatic Title Extraction approach to extract multi-level titles of each page.
- **Body Detection.** Another observation is that most pages contain large amount of noises, including navigational linkages, related topic linkages and so on. Filters are then built based on features of these areas, and the system is hence able to detect the body part of pages.

3. Similarity Matching.

- **Position Weighting.** Same query appearances in different positions of pages actually weigh differently. The probability of a page to be a key page in which the query appears in the beginning of its content is much higher than that of the page in which the query appears in the end. Illuminated by this, we use the position of query appearance in documents to boost their scores.
- **Query Combination.** In our system, each query is transformed into several forms: Phrase Query, Proximity Query, Ordinary Query, Expanded Query and Reshuffled Phrase Query. Each query weighs differently and may be applied to different fields of documents. Scores are finally merged together linearly. Query Expansion will be introduced in Section 3.4 of Expert Search Task.

For Expert Search Task, we adhere to the system we built last year[2], which focuses on Expert Annotation, query formation, two-stage similarity model development and enabling them in a statistical framework. This year we complement our system mainly in several aspects, among which, static ranking for expert candidates deserves underlining. Besides, data preprocessing as well as query expansion will also be introduced.

1. ExpertRank and Topic Sensitive ExpertRank. Compared to last year, the expert list is parsed from the corpus and thus may involve large noises. Hence we resort to static ranking method to handle this. The idea is to let experts vote for each other. A co-appearance can be viewed as a vote, similar to links in PageRank. In the light of this we proposed and experimented on ExpertRank and Topic Sensitive ExpertRank.

2. Data preprocessing.

- **Parsing Corpus for Expert Name List.** To get a list of experts, we parse email appearances in the corpus. Comprehensive situations like anti-spam and alias are handled. Names containing single word or host names are filtered.
- **VisualPageRank and Expert Homepage Detection.** Different HTML structures contribute differently in establishing evidences. We propose a DOM-Tree based VisualPageRank method to distinguish valuable and valueless HTML pages. Besides, we view name appearances in HTML Title as homepage signals, and boost the score of these pages as evidences to support an expert.

3. Query Expansion. We use narrative field of queries for query expansion. Features used to select words are: Query Similarity and Inverse Document Frequency.

The rest of the report is organized as follows. Section 2 and 3 records the technique highlights of Document Search and Expert Search tasks separately. Section 4 shows the preliminary experimental results of the submitted runs of both the two tasks.

2. Document Search

2.1 Static Ranking Approaches

For this part, firstly, we tried PageRank algorithm to compute the importance of each web page. However, the web graph is not complete in our corpus due to the sparseness of data. Thus PageRank does not perform well here. Similarly, Topic-Sensitive PageRank is not a good choice for this task, either. Then we tried HostRank to calculate web page importance.

HostRank algorithm starts from computing the importance of a host. And the hierarchy structure of the host is then used to distribute the host's importance to web pages within the host.

The HostRank algorithm consists of 2 steps:

- Calculate the Host's importance

Similar to PageRank algorithm, HostRank algorithm treats each host as a page in PageRank algorithm. Here we define hosts as the sub-layers of sub-portals like *"http://www.csiro.au/science"* , *"https://www.bioinformatics.csiro.au/GeneRave"*. This is because they are the highest level of URL whose importance should be distinguished.

- Propagation of the host's importance

After obtaining the importance of hosts, we can propagate the importance along the hierarchy of these hosts. Some factors should be taken into account when processing a page: depth of URL (indicating the inside linking structure); whether the page is an index page or a content page; links pointing to the page from outside hosts.

More specifically, if page P_i is pointed to by its ancestor page P_j in a hierarchy of a host, we have P_i's importance $HostRank(P_i)$ calculated by Equation (1).

$$HostRank(P_i) = \omega_i \cdot HostRank(P_j) \quad \textbf{(1)}$$

Where the weight ω_i is defined as:

$$\omega_i = \theta \cdot Link(P_i) + (1 - \theta)Index(P_i) \quad \textbf{(2)}$$

$Index(P_i)$ is a Boolean value denoting whether the page is an index page. We use rules to discriminate index pages. If the URL of a page contains "index" or "default", or end up with "/", it is viewed as an index page.

$Link(P_i)$ is defined as the percentage of the inlinks of page P_i, which is calculated in equation (3).

$$Link(P_i) = \beta \cdot \frac{OIL(P_i)}{\sum_{P_k \in Host\,(P_i)} OIL(P_k)} + (1 - \beta) \cdot \frac{IIL(P_i)}{\sum_{P_k \in Host\,(P_i)} IIL(P_k)} \quad \textbf{(3)}$$

Here $OIL(P_i)$ is the number of hyperlinks from pages outside the host to page P_i, and $IIL(P_i)$ is the number of hyperlinks from inside the host to the page P_i. $Host(P_i)$ is the host containing page P_i.

2.2 Data Preprocessing

2.2.1 Title Extraction

Title field of web documents has been proved to be a high-quality information source in web search Tracks at TREC. From observation, half of the documents in our corpus lacks the title field or leave the field as "untitled". Therefore, we extract not only the <TITLE> tag of html pages but also the titles from the body content by exploring <H1>, <H2> tags. Then a Multi-Level Title field is obtained.

From the preliminary results of our evaluation over test query, using the VSM model to perform search on title field can improve the performance dramatically.

2.2.2 Body Detection

Here Bodies of documents are defined as the main content of a document, which is the most important information source of a page. Through our observation, body fields usually contain head tags like <H1>, <H2>…<H6>, while non-body fields like navigator or templates usually have a higher percentage of links. Thus we develop our 2-step algorithm as follows.

- Dividing the page based on DOM tree structure.
    1. If the length of text of a node is 0, ignore this node.
    2. If the maximum length of text of a node's children is less than 30% of the page, then keep it as a part.
    3. Divide other DOM nodes.
- Filtering divided parts.
    1. If a head tag is found in a part, then keep this part.
    2. If no head tags are found and the percentage of link text to the total text is greater than 70%, delete this part.
    3. Keep other parts.

By this method, many navigators and useless templates have been removed, while most of the useful information has been kept.

2.3 Similarity Matching

2.3.1 Position Weighting

Sometimes a page contains certain relevant information about a topic, but it is not a key page to the topic (e.g. News Summaries). The position of query terms' appearances in a page is a good feature to judge whether this is key page or not. Thus we exploit Position Weighting for queries and documents.

For query *Q* and document *D*, the Position Weight can be calculated as:

$$PW(Q,D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{Len(D)}{Pos(q_i, D)} \quad (4)$$

Here $|Q|$ is the number of different query terms in Q, $Pos(q_i, D)$ is the first appearing position of query term q_i in D and $Len(D)$ is the length of D. This score is finally combined to the Similarity Score after scaling.

2.3.2 Query Combination.

Query preprocessing is adopted in our system. Each query is transformed into several forms: Phrase Query, Proximity Query, Ordinary Query, Expanded Query and Reshuffled Phrase Query. Each query form weighs differently and may be applied to different fields of documents. Query Expansion technique will be introduced in Section 3.4; Reshuffled Phrase Query denotes the Phrase Query formed by reshuffling the query terms; others are similar to [2].

Our system uses the OKAPI BM25 [3] model to compute the Similarity Score between queries and documents' title, keywords, anchor text, H1, H2 and extracted body field respectively. Each field is calculated under five kinds of query formats mentioned above. The scores are combined linearly after normalization.

In order to achieve a combinable Similarity Score for different queries and document fields, we normalize each score into a value between [0, 1]. All the BM25 scores are obtained by using a form of standard BM25 formula with parameters k1 set to 1.2 and k2 to 20.

$$S_{bm25}(Q,D) = \sum_{q_i \in Q} IDF(q_i) \cdot \frac{TF(q_i, D) \cdot (k_1 + 1)}{TF(q_i, D) + k_1 \cdot (1 - k_2 + k_2 \cdot \frac{Len(D)}{AvgLen(D)}} \quad (5)$$

We use the normalization introduced by [4].

After obtaining the combined score, we use HostRank score to re-rank the documents. The re-ranking formula is:

$$S_{rerank}\ (Q,D) = S_{bm25_{normalized}}\ (Q,D) \cdot PW(Q,D) \cdot (1 + \alpha \cdot HostRank(D)) \quad (6)$$

Where $S_{bm25_{normalized}}\ (Q,D)$ is the combined BM25 score and $HostRank(D)$ is the HostRank score computed within the corpus.

3. Expert Search

3.1 ExpertRank and Topic Sensitive ExpertRank

Compared to last year, the expert list is parsed from the corpus and thus may involve large noises. Hence we resort to static ranking method to handle this.

The idea of ExpertRank is to let experts vote for each other. A co-occurrence of two experts would contribute two directed links between them, one pointing from the first expert to the second and the other from the second back to the first. After this a directed graph is constructed, of which each vertex represents an expert candidate and each edge indicates a certain propagation of importance from the source to the target node. Then we can apply the ExpertRank algorithm to the expert graph. The algorithm is basically the same with PageRank algorithm, but different in the sense that here edges from one expert candidate to another are weighted. This is because two experts co-occurring regularly are meant to share much more importance with each other than with others. Below we describe the algorithm in detail. Some of the concepts and formulas are derived from [1].

3.1.1 Expert Rank

Consider expert candidate *c* and *c'* pointing to c, *P(c)* represents the set of all possible *c'*. Let *W(c',c)* be the weight of the edge from *c'* to *c*, and *W(c')* be the sum of weights of outgoing edges of *c'*. Also the rank value of candidate *c* is regarded as *Rank(c)*. For the initialization, *Rank(c)* for every candidate is set to 1/*N*, where *N* is the total number of expert candidates. Then the iteration of propagation is started until convergence. In each iteration i, the rank of each candidate *c* is re-computed as equation (7):

$$Rank_i(c) = \sum_{c' \in P(c)} Rank_{i-1}(c') \cdot W(c',c)/W(c') \qquad \textbf{(7)}$$

The computation can also be explained as eigenvector calculation as expressed by [1]:

$$\overrightarrow{Rank} = M \times \overrightarrow{Rank} \qquad \textbf{(8)}$$

From this point of view, we did not really do the iteration in practice. Since the expert graph we built is symmetrical, it is easy to solve equation (8). The solution is:

$$Rank(c) = \sum_{c' \neq c} \sum_{d \in D} cooccur(c, c') \qquad \textbf{(9)}$$

Here D is the entire document collection. $cooccur(c, c')$ is 1 if c and c' co-occur in document d. After normalizing $\|\overrightarrow{Rank}\|_1$ to 1, it yields the final ExpertRank.

$$Rank_{normalized}\ (c) = \frac{Rank(c)}{\|\overrightarrow{Rank}\|_1} \qquad \textbf{(10)}$$

3.1.2 Topic Sensitive ExpertRank

The intention of Topic Sensitive ExpertRank is that, the distributions of expert importance over different topics are not similar, and it's unreasonable to compare the importance of experts in different fields. Thus the rank value calculated by the global propagation is very rough. To perform more accurate statistic ranking and better approximate the importance of the expertise of experts, we clustered the entire corpus into 70 topic related categories by URL analysis. The clustering method is to first build a suffix tree of all URL hosts (together 100 in all) in the CSIRO corpus, and cut the tree at level 3.

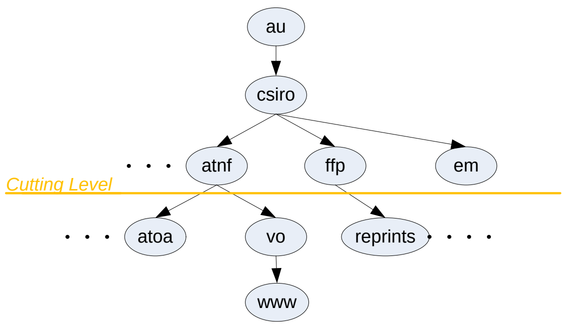

Fig. 1. A Sample of URL Suffix Tree

For each category (topic) T_k, we bias the ExpertRank vectors using equation (11).

$$Rank_k(c) = \sum_{c' \neq c} \sum_{d \in T_k} cooccur(c, c') \qquad \textbf{(11)}$$

By performing ExpertRank algorithm on each category k, we obtain 70 ExpertRank vectors biased by 70 different topics.

Besides, the term vector of each category is calculated and indexed. The real time computation of an expert candidate's final rank value involves the conditional probability $P(T_k|q)$, where q is the given query. $P(T_k|q)$ is calculated using equation (12).

$$P(T_k|q) = \frac{P(T_k) \cdot P(q|T_k)}{P(q)} \propto P(T_k) \cdot \prod_i P(q_i|T_k) \qquad \textbf{(12)}$$

Here q_i is the ith term of query q. In practices we use Query Expansion to better approximate the distribution of queries over the vocabulary.

The final rank value of candidate c, *Rank(c)*, is computed by equation (13).

$$Rank(c) = \sum_k P(T_k|q) \cdot Rank_k(c) \qquad \textbf{(13)}$$

The Topic Sensitive ExpertRank is later multiplied to the similarity ranking value to yield the score of expert candidates. In the submitted runs we use Topic Sensitive ExpertRank instead ExpertRank.

3.2 Data Preprocessing

3.2.1 Parsing Corpus for Expert Name List

Using the full name as email address is recommended in many enterprises for easily management. Benefited from this, we match emails to expert names by extracting them directly from the email address. Two problems calls for attention in the process of parsing email appearance. The first is how to handle anti-spam formats while the second is how to recognize the emails that are not dedicated for a single person.

Table 1. Example of anti-spam formats

Email	Anti-Spam Format
Tom.McGinness@csiro.au	Tom . McGinness @ csiro . au
	Tom DOT McGinness AT csiro DOT au
	firstname.lastname@csiro.au (Tom McGinness shows in the same page.)

Our method firstly replaces all the tokens such as "dot", "at" and "atmark" by the corresponding symbols. Then we use an enhanced regular expression to detect all the emails from context. We also extract emails containing "first" as a substring, such as firstname.lastname and firstname.surname. Named Entity Recognization Tool is used to extract all the names in the context of these emails to complete the candidate list.

To handle the problem of non-personal emails and build a clean expert name list, we construct several filtering rules:

- Filter out emails that do not end with the host name of the organization.
- Filter out emails containing numbers.
- Filter out emails with host names appearing in the person name part. Host names are extracted from the URL hosts. (e.g. If URL http://www.bio.csiro.au/* exists in the corpus, the email xxx.bio@csiro.au should be filtered.)
- Filter out emails with single letter in its person name part.

After this a list of expert candidates (both names and their corresponding emails) is obtained, and we are able to perform the expert identification algorithm[2] we proposed last year to index all the appearances of expert candidates in the corpus.

3.2.2 VisualPageRank and Expert Homepage Detection

The intention of VisualPageRank is to recognize and degrade pages that are unhelpful or too noisy to establish a good evidence for expert search, while Expert Homepage Detection contributes in the opposite way by detecting and upgrading highly possible expert homepages to support the correspondent experts.

Through observation we find that unhelpful or noisy pages come from either too simple or too complicated pages. So we explore into the HTML structure of web pages. A Vision based content tree is built by separating texts using a group of tags.

Thresholds are set to the number of blocks and total depth of the tree. By degrading the score of pages that are below the lower threshold or above the upper threshold, we are able to reduce their influences to the task.

For Expert Homepage Detection, since we only want to utilize the highly possible expert homepages, we simply run the expert name identification algorithm[2] on title field of pages, and filter out pages that contain only low confidence name masks[2] or multiple expert candidates. The left pages are viewed as the homepage to the corresponding expert in its title, and their support to the homepage owner are boosted.

Note that the score of a page here denotes to Q_d in [2].

3.3 Query Expension

As mentioned above, to better approximate the distribution of queries over term vocabulary, we adopt Query Expansion in our system. Query Expansion is used in two ways in the system, the first is to serve as one additional query format for the computation of Q_q in [2], the other is for calculating $P(T_k|q)$ in equation (8).

The candidate terms of Query Expansion comes from the narrative fields of given topics. There are two approaches for selecting terms. The first is quite straight forward, by calculating the inverse document frequency of each word. Three terms with the highest score are added into the original query term bag. The second approach we try is to extract a window based context vector in the corpus for each candidate term, and calculate the cosine similarity between the context vector and the query vector. Also three words with the highest score are then selected. The idea of this approach is to keep the topic of the query focused during the expansion. Finally the first is adopted among the final runs, since the window size for context extraction is very hard to determine lacking training data.

4. Experiment Result

Here we list the evaluation results of the 4 Document Search runs and 4 Expert Search runs we submitted from TREC committee. Table 2 and Table 3 show the results of Document Search and Expert Search respectively.

Table 2. Evaluation of Document Search Runs

Run ID	DS-BASE	RS-PQ	STEM	MAP	R-PREC
SJTUEntDS01	Y	N	N	0.3130	0.3365
SJTUEntDS02	Y	Y	N	0.3793	0.3934
SJTUEntDS03	Y	N	Y	0.3222	0.3395
SJTUEntDS04	Y	N	N	0.3295	0.3454

Here DS-BASE includes all the techniques we mentioned in Section 2 except Reshuffled Phrase Query (denoted here as RS-PQ). STEM stands for stemming of words before indexing. Run SJTUEntDS04 is different from SJTUEntDS01 since it's a feedback run which uses given key pages to perform further Query Expansion.

Table 3. Evaluation of Expert Search Runs

Run ID	ES-BASE	TS-ER	HP-DT	MAP	MRR
SJTUEntES01	Y	N	N	0.4395	0.6140
SJTUEntES02	Y	Y	N	0.4343	0.6039
SJTUEntES03	Y	N	Y	0.4427	0.6131
SJTUEntES04	Y	Y	Y	0.4410	0.6146

Here ES-BASE includes the Email Parsing, VisualPageRank, Query Expansion and the system last year[2] excluding Cluster Based Re-ranking. The exclusion is because the clustering of automatically parsed expert list is very inaccurate compared to the predefined list last year. TS-ER stands for Topic Sensitive ExpertRank while HP-DT refers to Homepage Detection.

5. Reference

[1] T. Haveliwala: Topic-Sensitive PageRank. In: *Proceedings of WWW2002*, 2002.

[2] S. Bao, H. Duan, Q. Zhou, M. Xiong, Y. Cao and Y. Yu: Research on Expert Search at Enterprise Track of TREC 2006. In: *proceedings of 15th Text Retrieval Conference (TREC 2006)*, 2006.

[3] S.E. Robertson, S. Walker, M. Hancock-Beaulieu, M. Gatford: Okapi at TREC. In:*Text REtrieval Conference*.

[4] R. Song, J.-R. Wen, S. Shi, G. Xin, T.-Y. Liu, T. Qin, X. Zheng, J. Zhang, G. Xue, W.-Y. Ma: Microsoft Research Asia at Web Track and Terabyte Track of TREC. In: *proceedings of the Thirteenth Text Retrieval Conference Proceedings (TREC-2004)*, 2004.

[5] G. Xue, Q. Yang, H. Zeng, Y. Yu, Z. Chen: Exploiting the Hierarchical Structure for Link Analysis. In: *Proceedings of SIGIR2005*, 2005.

[6] L. Page, S. Brin, R. Motwani, T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. Paper SIDL-WP-1999-0120 (version of 11/11/1999).