Ricoh Research at TREC 2006: Enterprise Track

Ganmei You, Yaojie Lu, Gang Li, Yueyan Yin
Ricoh Software Research Center Beijing Co., Ltd, Beijing, China
{ganmei.you, yaojie.lu, gang.li, yueyan.yin}@srcb.ricoh.com

1. Abstract

This article presents our contributions to expert search and discussion search of Enterprise Track in TREC 2006. In discussion search, we take advantage of the redundant patterns of emails, such as the subject, author, sent time, etc., which we incorporate in a field-based weighting method to mine discussion topics with more robustness. Some non-content features, such as time-line and mail thread are found to be useful as experiments showed they improve the precision of the search.
In expert search, two variants of the BM25 and DFR_BM25 weighting models - namely V-BM25 and V-DFR_BM25 - are put forward. Query-based document length, not profile length, is used as document length in these weighting models to eliminate multiple topic drift. In addition, we propose a variant of an existing phrase weighting model to decrease topic confusion (V-phrase) and a two-stage field-based search method to refine the results. We demonstrate these approaches can effectively improve expert search.

2. Discussion Search

In this section, related work is presented firstly; secondly data cleaning and feature extraction is introduced; thirdly terminology and methods we used are discussed; at last submitted runs are listed.

2.1 Related work

In [1], Anh *et al*. (Melbourne University) made a baseline run on an index from which all quoted text had been stripped. In this baseline run, the document scores were then supplemented, first by scores from a parallel index of the quoted text, then by scores of other messages in the same thread, and finally by the frequency with which the message's author is posted to the W3C mailing lists. A separate run was made using an impact-based system. The results showed the impact-ordered run is superior to the chosen baseline, and retaining quoted text is superior to removing it. Enhancing document scores with thread information is a promising technique.
Craswell *et al.* (Microsoft Cambridge) identified three text fields: subject, body and quotation [2] and treated each of these differently over a uniformly weighted baseline. Their results emphasize the importance of having appropriate training data (lacking for discussion search) to get satisfying results.
Vechtomova *et al.* (University of Waterloo) also adopted thread properties to identify the discussions [3]. All the test (pseudo-relevance feedback, and use of some structure information) runs of CSIRO and ANU performed poorer than the base run that simply ignored email structure and treated all elements equally.

Maybe there are bigger gains to be made by considering email-specific features like thread structure [4].

2.2 Data cleaning and feature extraction

Figure 1 shows the typical structure of a discussion message. We can divide it into 19 parts (A-N), from each of which information can be extracted.

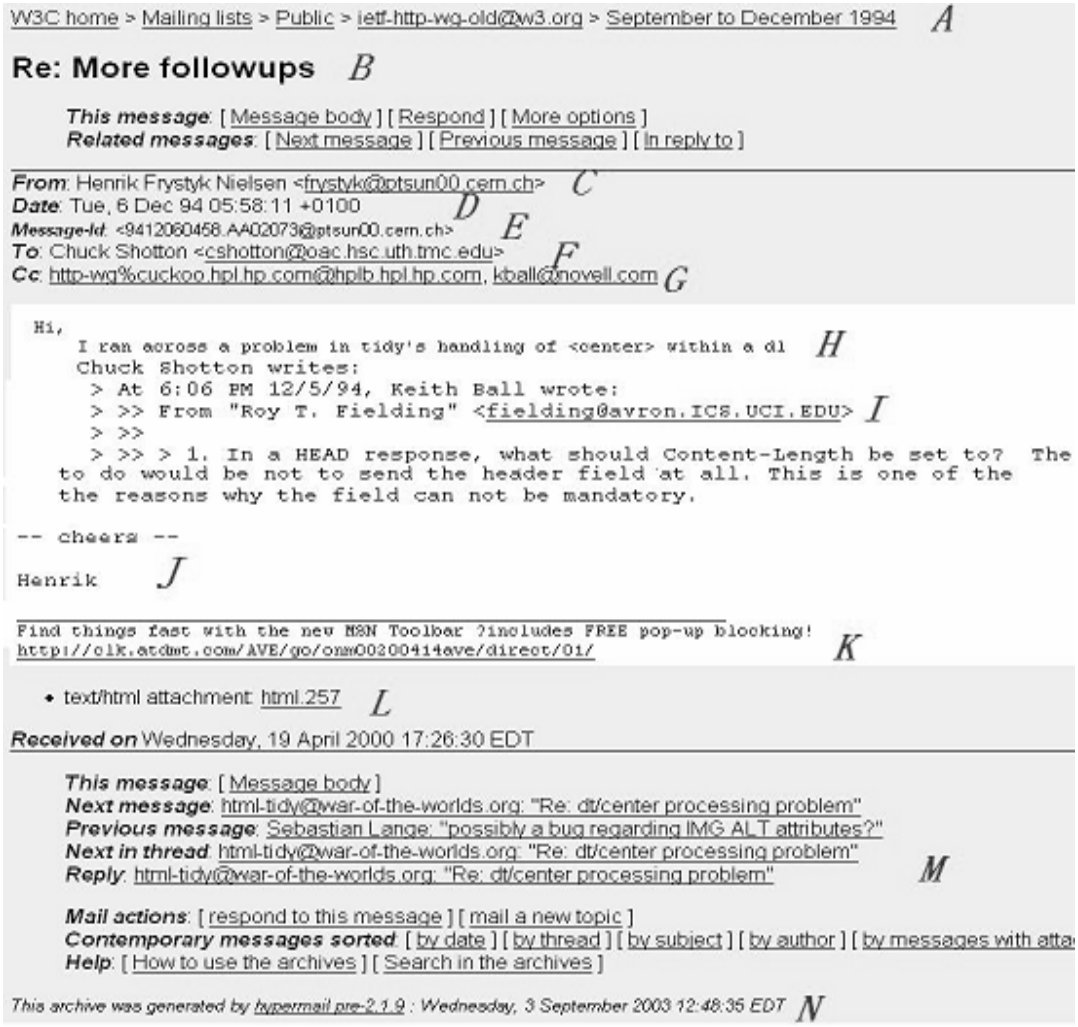

Figure 1: Discussion thread extracted from the W3C corpus.

The distinction between these parts - and their role as content-related or non-content-related - is the cornerstone of our discussion retrieval scheme. According to the HTML tag information and some structure feature (e.g. quotations are identified by the quotation character '>') we can easily divide the document into parts labeled from A to N in figure 1.

Part A is the navigation field;

Part B is the subject of the message. It is a summary of the key points of the message and therefore an important piece of information;

Part C is the author of the message;

Part D is the creation time of the message;

Part E is the unique ID of the message;

Parts F/G can define the category of the message and show the relationship of the author with other members;

Parts H/I/J/K refer to the main content of the message. Parts J and K are the greeting part of the message and the advertisement section respectively. These parts have no relationship with the content, so we neglected them. The quotations (part I) - which are passages of the original text - are identified by quotation characters that prefix each line. Such quotation characters can be '>' or ':' or probably other characters or regular expressions; combinations of them usually define the quotation depth. The new part (Part H) contains the new content typed by the author of the message;

Part L is the attachment part;

Part M is the thread information from which we can extract the reply chain;

Part N is also a useless part.

2.3 Terminology

- Timeline**:**

  In this paper, we call *timeline* the time elapsed between the reception of an e-mail and its next reply in the discussion.

- Thread

  Some e-mails often reply to a certain e-mail, and usually discuss a single topic. In such situations, all e-mails that discuss one certain topic belong to one *thread*. The first e-mail introducing the topic is the root of the thread, and the e-mails that answer this e-mail are the children of the thread root. Note that an e-mail with no reply corresponds to a thread containing a single element.

- Field

  One e-mail has its author, email address, and other attributes like: send time, receivers, etc. These features are called *fields* of the e-mail in our paper.

2.4 Methods

2.4.1Timeline-based approach

It is highly probable that the e-mails that discuss a certain topic are sent within a reasonably short timeline. Conversely, "noise e-mails" (e-mails irrelevant to the topic) occur more randomly in time so that the timeline can filter them out in the search process. The timeline-based filtering method can be decomposed into the following steps:

- Get top N e-mails (N = 5 ~ 30) from the first search results.
- Sort them by the send time (optional).
- Group these e-mails by the send time
- For each group:

  → Get the <start, end> time slot

  → For each e-mail in the top 1000 results:

  → If its send time is in the slot, add the constant *c* to its score

One key point of this method is how to group e-mails; figure 2 shows how it works:

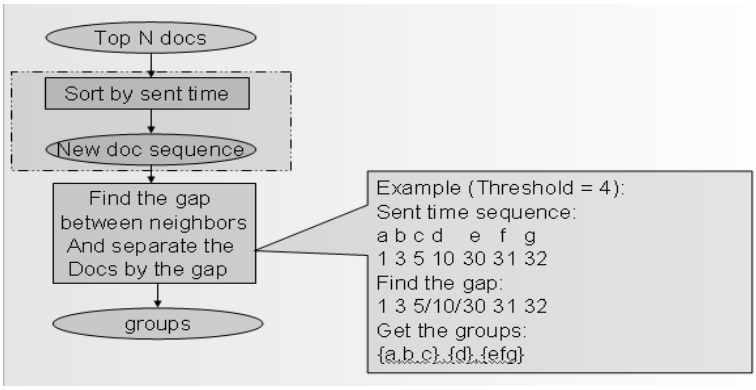

Figure2: Workflow chart of timeline-based method

2.4.2 Thread-based approach

This method uses the thread information to cluster the top N e-mails and adjusts their scores according to the groups they belong to in order to get rid of noisy messages. It does not care about how to search the results; it just focuses on re-ranking the found results by adjusting their scores.

Then how to adjust the scores? We define what we call the "distance" in thread, which is the space between two e-mails in the same thread. We represent a thread by the following tree:

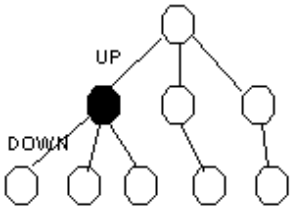

Figure 3: Tree-based representation of a thread

From the black e-mail node (see figure 3), the upper one is the e-mail that it replied to; the lower ones are the e-mails that replied to the black e-mail. Then, the distance between the black one and the upper one is equal to UP (it is a parameter whose value ranges from 0 to 2 in our system), the distance between the black one and the lower one is equal to DOWN (parameter which value is 1).

The thread-based approach goes through every e-mail in the top N results. For each e-mail, we first obtain the thread it belongs to; secondly, we compute the distance from this e-mail to every other e-mail in the same thread as follows:

- Initialization of parameters: DOWN = 1; UP : 0 ~ 2
- From the seed node:
- use the stack to compute:
- Push the parent & children into the stack with distance:
  Parent: this node's distance + UP,
  Children: this node's distance + DOWN
- Pop the node in the stack, and push its parent & children upper. Stop until the stack is

empty

Next, we use the computed distance to adjust every e-mail's score: if the average distance to the group is lower than a threshold, the score of this e-mail will be increased, otherwise, the score will be reduced. At the end, every e-mail has a new score that is used to rank them again. After re-ranking, some correct answers may have a higher relevance rank than without such processing, and some wrong answers may be pushed to lower positions.

2.4.3 Timeline & thread-based method

This method uses the timeline to compute the distance in one thread, and this is the main difference from the method above. In this case, the distance is the timeline. The other steps are similar to the ones of the method in 2.4.2.

2.4.4 Advanced field-based method

As we know, the different parts of the document have a different importance. The important part of a document contains the terms that are likely to match the search query. The proposed advanced field-based search method takes advantage of such observation to improve the retrieval results. However, note that a part considered as important in one document can be seen as less important in another. For instance, the *subject* part is very important as it contains the topic of the discussion, but when it is replied, the original subject will be prefixed with 'Re:', adding no information to the content. Besides, the author who replies may type text with a new content that the subject cannot incarnate. In our experiments, we do not fix the weight of each field, but consider many interdependent fields together. For example, if a term appears in both the subject and body, we will set a high weight to the subject, but when only the subject contains the term, the weight of subject will not get a very high weight.

2.4.5 Query Expansion

We used two kinds of query expansion in our experiments. One relates to the pseudo-feedback and the other is called 'expanding term form narrative part'.

- Pseudo-feedback: we check whether the top N documents in the search results are relevant and we extract some terms from these documents to carry out the expansion.
- Extract related terms form the narrative part: because the narrative part of the query is very long and there are many noisy terms, we do not extract all terms directly but only the useful ones relevant to the query part to achieve the expansion.

2.5 Submitted Runs

There are 5 runs in our discussion search.

- Run 1, timeline + field-based +query expansion

This method uses the first search results to extract query terms for the expansion and searches again. Then it uses the timeline and fields to optimize the search results.

- Run 2, timeline + field-based +query expansion and double write abbreviations

Before the search, duplicate abbreviations in the corpus like: "RDF" to→ RDFRDF, where RDF stands for "resource description framework". Then we use the first search results to extract query terms for the expansion and search again. Then we use the timeline and fields to optimize the search results.

- Run 3, timeline + query expansion and double write abbreviations

Before the search, duplicate abbreviations in the corpus like: “RDF” to→ RDFRDF, where RDF stands for “resource description framework”. Then we use the first search results to extract query terms for the expansion and search again, then use the timeline to optimize the search results.

- Run 4, field-based + query expansion and double write abbreviations

Before the search, duplicate abbreviations in the corpus like: “RDF” to→ RDFRDF, where RDF stands for “resource description framework”. Then we use the first search results to extract query terms for the expansion and search again. Then use the field-based approach to optimize the search results.

- Run 5, timeline + advanced field-based method + query expansion with narrative part

This method extracts query terms from the narrative part for the query expansion and searches again. Then, we use the timeline and field-based approaches to optimize the search results. Tables 1 and 2 show the precision results of the five different runs while considering different levels of relevance.

Table 1 Precision results of the five runs

(Scores are computed where judging levels '2'* (contains a pro/con) and above are considered relevant.)

	run1	run2	run3	run4	run5
num_ret	4776				
num_rel_ret	3827	3901	3908	3924	3850
MAP	0.2648	0.2681	0.2749	0.2593	0.2852
P5	0.4043	0.3826	0.3783	0.3348	0.4478
P10	0.387	0.3565	0.3739	0.3283	0.437
P15	0.371	0.3536	0.3681	0.3174	0.4101
P20	0.3598	0.3598	0.3587	0.3174	0.3913
P30	0.3355	0.3297	0.3341	0.3058	0.3659
P100	0.2763	0.2837	0.2861	0.2733	0.2917
P200	0.2278	0.2328	0.2416	0.2312	0.2351
P500	0.1373	0.1417	0.1455	0.1419	0.1413
P1000	0.0832	0.0848	0.085	0.0853	0.0837

Table 2 Precision results of the five runs

(Scores are computed where judging levels '1'* (relevant to the topic) and above are considered relevant.)

	run1	run2	run3	run4	run5
num_ret					4776
num_rel_ret	3827	3901	3908	3924	3850
MAP	0.3999	0.397	0.4026	0.3855	0.4065
P5	0.628	0.6	0.6	0.596	0.652
P10	0.582	0.582	0.578	0.57	0.61
P15	0.5587	0.5587	0.548	0.5413	0.58
P20	0.544	0.543	0.525	0.531	0.561
P30	0.5147	0.512	0.498	0.5053	0.5307

P100	0.4408	0.4436	0.4478	0.438	0.4452
P200	0.3565	0.3617	0.3704	0.3641	0.3612
P500	0.2149	0.2194	0.2242	0.2203	0.2154
P1000	0.1274	0.1295	0.1298	0.1303	0.1262

*

0: not relevant.

1: relevant, does not contain a pro/con argument.

2: relevant, contains a negative (con) argument.

3: relevant, contains both pro and con arguments.

4: relevant, contains a positive (pro) argument.

From the results, we notice that the traditional field-based method has little effect on the MAP (see table 1). It might be due to the numerous extracted fields from the corpus so that there are too many parameters in this method. As a result, there is not enough training data to tune these parameters, which explains why the method is not so effective.

Besides, we see that the timeline-based method effects efficiently the MAP. It proves that using the timeline to optimize the results is reasonable (see table 2).

3. Expert Search

Expert search is a new sub-task in TREC Enterprise Track. Given a topic, the task is to identify who are experts on the topic. It is a useful and attractive research because there are very similar requirements in enterprises.

Expert search is not a simple task because we cannot apply classical Information Retrieval (IR) models directly to get the results. For instance, methods solely based on keywords cannot achieve good results, thus new solutions are wanted. Two common search methods are easy to find out. The first one is document search and voting. We first search documents relevant to a given topic using a classical IR model, then we sort the experts based on their occurrence frequency in the documents relevant to the topic. The other approach is profile search. That is, we first process the corpus and build a profile for each expert. Then, we can use classical IR models to find experts in the profiles for each topic. We tested the two methods with TREC 2005 data. Experiments proved that the latter method can achieve about 20% higher Mean Average Precision (MAP) than the former one. Therefore, we have adopted the profile search method in our expert search scheme.

In this section, firstly related work is introduced; then our pre-processing of the data and search methods are presented; finally runs we submitted are discussed.

3.1 Related work

Fu *et al.* (Tsinghua University) used a “document reorganization” method that is effective for the expert finding task [5]. It reorganizes the descriptions from all sources of information for each candidate expert by

allocating different weights to the documents' sources of context and ranking. Such approach yields better performance than treating all sources of information in the same way. Finally, a bi-gram retrieval method increases the precision of the expert search.

Cao *et al.* (Microsoft Research Asia) used a two-stage language model and window-based co-occurrence sub-model [6]. They use metadata in building co-occurrence models and a clustering-based re-ranking method.

Yao *et al.* (Peking University) jointly used three methods to achieve the search [7]: a traditional IR technique, an e-mail clustering method and an entry page finding scheme. The authors used two-result aggregation methods of linear synthesis and Markov chain to combine the three generated results. Experiments demonstrate that the traditional IR method is useful if the query is well generated. The e-mail clustering method is effective when the mailing list is relevant to a unique work group or committee, and the entry page finding method is valuable when the topic is the theme of a special group.

MacDonald *et al.* (Glasgow University) created candidates profiles and used the expC2 DFR weighting model to rank them [8].

Ru *et al.* (Beijing University of Posts and Telecommunications) used three methods: a two-stage ranking (BM25 weighting model and a language model based on KL-divergence to rank documents), a corpus refinement and a name disambiguation [9].

3.2 Pre-processing of the data

To create a profile for each expert, we must first process the corpus. The W3C corpus consists of web pages, emails, WIKI pages, CVS data and text files. Our goal is to find all related information for each expert and create his/her profile database. The processing steps are as follows:

3.2.1 Create a candidate identifier

In the expert list, a full name and at least one e-mail address is provided for each expert. However, experts will not always appear in the form of a full name or an email address. So we propose to add anchor texts to the candidate identifiers list. The anchor text is a hyper-link pointing to the email address of a candidate that we obtain by scanning each web page.

3.2.2 Find relations between candidates and documents

We visit each page to find experts by using the candidate identifier in each document and record the occurrence information. In this step, we use the Wu-Manber algorithm to do multi-pattern matching. The occurrence information is recorded in a XML file.

3.2.3 Extract candidate occurrence information

We extract the occurrence information based on the relations we have built. The occurrence information is obtained within a window - of at most 50 words in size – centered on each occurrence position of the expert.

3.2.4 Extract Web page core

Important information is extracted from each Web page that has candidate information. We extract title, headers, abstract, text occurrence, keywords and descriptions from HTML metadata. Headers include any level HTML header, such as <h1>, <h2>. The abstract is the first paragraph below headers entitled "abstract". Occurrence text is fifty words before and after each candidate identifier. We call this important information the Web page core. Only the Web page core is used in expert profiling but not the whole page

content.

3.2.5 Remove duplicated Web pages

We found that there are some duplicated Web pages in W3C corpus that wrongly enhance the information of the repeated topics. These duplicated Web pages introduce a bias in the search results from this W3C data collection, thus decreasing the MAP. We use two methods to remove duplicated Web pages. One is by URL. We found some specific URL patterns are duplicated, such as in http://esw.w3.org/topic/Algae?action=diff&date=1059952621. URLs which have different date parameters but similar contents are considered as duplicate. The other removes a duplicated Web page when the repeated part of each field (such as title, header, occurrence information, etc.) exceeds a proportion of the total length of this field.

3.2.6 Generate profile database

We merge all non-duplicated Web page cores relevant to an expert into his/her profile. In our experiments we got 696 profiles. However, 396 other experts couldn't be retrieved in this pre-processing phase.

3.3 Search methods

3.3.1 Two-stage search method

We propose a two-stage search method for expert search. In this method, a new weighting model and a new phrase search method are combined together.

3.3.1.1 Variants of weighting models

In expert search, all queries that are titles are short. So the BM25 weighting model [10] is adopted. The BM25 we used calculates the relevance weighting model score of a profile *d* for a query *Q* by the following formula:

$$w\_score(d,Q) = \sum_{t \in Q} \frac{tf}{K + tf} \frac{qtf}{qtf + k_3} \log(k_2 \frac{N}{N_t} + 1.0) \qquad (3.3.1)$$

Where *tf* is the frequency of term *t* in the document (profile) *d*; *qtf* is the query term frequency; *N* is the number of documents (or profiles) in the whole collection; *Nt* is the document frequency of term *t*; *k2* and *k3* are parameters. *K* is defined as:

$$K = k_1((1-b) + b\frac{l}{avg\_l}) \qquad (3.3.2)$$

Where *l* and *avg_l* are the document length and the average document length in the collection respectively; *k1* and *b* are parameters.

Note that in equation (3.3.1), we added a unit constant inside the *log(.)* function to ensure that the score is always positive. As a result, all profiles relevant to any term of the query can be found, which increases the recall ratio.

To increase the precision, we use the DFR_BM25 weighting model [11]. In this model, the relevant weighting model score of a document *d* for a query *Q* is given by:

$$w\_score(d,Q) = \sum_{t \in Q} \frac{TF}{TF + k_1} \frac{(k_3 + 1)qtf}{k_3 + qtf} \log(\frac{N - N_t + 0.5}{k_2 N_t + 0.5}) \qquad (3.3.3)$$

Where *qtf*, *N* and *Nt* have the same meaning as those in equation (3.3.1); *k1*, *k2* and *k3* are parameters. *TF* is

$$TF = tf \log(1.0 + c\frac{avg\_l}{l}) \tag{3.3.4}$$

Where *tf*, *avg_l* and *l* have the same meaning as those in formula (3.3.1) and (3.3.2); *c* is a parameter.

In equation (3.3.3), $\log(\frac{N - N_t + 0.5}{k_2 N_t + 0.5}) \le 0$ if *Nt* > *N*/ (*k2*+1). That is, if more than a pre-set percentage (1/ (*k2* +1)) of profiles contains a term, the term is called a frequent term. In the model, frequent terms of a query are ignored since frequent terms have weaker document differentiation ability. For example, we can consider as frequent a term that is contained by more than half of profiles. Such frequent terms have no contribution to the document score.

In our experiments, we search profiles to find experts. In other words, profiles are seen as searched documents. These profiles are created by the above data processing. Because each profile is formed from multiple Web pages related to different topics, each profile may contain multiple topics. We experimentally verified this fact. When the profile length is used as document length in BM25 or DRF_BM25 weighting model, some experts rank very low because they are interested in several fields, which results in very long profile lengths. In other words, those profiles have multiple topic noise besides the given topic. To avoid such problem, the query-based document length instead of the profile length is used as document length in BM25 and DRF_BM25 weighting models. Query-based document length is got in this way: for a query, Web page cores are retrieved and then relevant Web page cores are found; for each profile, the query-based document length is the sum of the document lengths of relevant Web page cores matching the profile's candidate. In other words, only the length of parts of the profile that are relevant to the query are used to compute the document length. As a result, the model exposed in equation (3.3.1) can be regarded as a variant of the BM25 weighting model (V-BM25) and formula (3.3.3) is a variant of the DFR_BM25 weighting model (V- DFR_BM25). Our experiments prove that the average precision of the variants of the weighting models is higher than that of classic weighting models. Document scores computed by the weighting model will be adjusted by the following phrase score.

3.3.1.2 Variant of phrase weighting model (V-phrase)

Phrase search consists in seeking k-adjacent terms of a given query in documents. In the method of section 3.3.1.1, the document (or profile) score is a linear combination of the query terms. That is, only individual terms are considered while terms relationships are ignored. Such approach may be sensitive to topic drift. For example, consider the expert candidate 0190 in semantic Web coordination who will have interactions with other experts in a distinct field such as P3P. If the query is "Semantic Web Coordination" and only classic weighting models are used, the candidate 0190 ranks high even though he is not an expert in Semantic Web Coordination. However, when we search for relevant profiles with adjacent terms, such as "Semantic Web", "Web Coordination" or "Semantic Web Coordination", the candidate does not rank No.1. It illustrates how topic drift decreases the efficiency of such phrase search approach.

The relevant document phrase score for a query *Q* is given by:

$$p\_score(d, Q) = \sum_{P \in Q} \frac{PF}{PF + k_1} \frac{(k_3 + 1)qpf}{k_3 + qpf} \log(\frac{N - N_p + 0.5}{k_2 N_p + 0.5}) \tag{3.3.5}$$

$$PF = k * pf \log(1.0 + c\frac{avg\_l}{l}) \quad (3.3.6)$$

Let |...| indicate the term number. *P* is a k-adjacent phrase of query *Q*, or |*P*| = *k*; *pf* is the frequency of phrase *P*. *qpf* is the phrase frequency in query *Q*. *N* is the number of documents (or profiles) in the whole collection; *Np* is the document frequency of phrase *P*; *k1*, *k2* and *k3* are parameters. *l* is the query-based document length. The number of k-adjacent phrase terms is |*Q*| - |*P*|+1.

3.3.1.3 Document (profile) score

The document (profile) score is the linear combination of the document weighting model score and the phrase score:

$$score(d,Q) = w\_score(d,Q) + k_4 * p\_score(d,Q) \quad (3.3.7)$$

3.3.2 Two-stage field-based search method

As was described in section 3.2, core information (title, abstract, headings and window information) is extracted from each Web page. Then, for each candidate, titles of all related Web pages compose the title profile. In this way, candidate information consists of four fields: title profile, abstract profile, headings profile and window information profile. A two-stage search method is applied to each field. Each candidate profile score is a linear combination of field-based profile scores:

$$score(c,Q) = \sum_{f \in fields} \lambda_f score(f,Q) = \sum_{f \in fields} \lambda_f (w\_score(f,Q) + k_4 * p\_score(f,Q)) \quad (3.3.8)$$

Where $fields = \{title\ \ profile, abstract\ \ profile, heading\ \ profile, window\ \ profile\}$,

λ_f and k_4 are parameters.

3.4 Submitted Runs

We submitted five runs of expert search in TREC2006. All five runs are based on the same data processing method as exposed above and with different weighting methods and parameters. The major differences between the five runs are described below:

1. SRCBEX1 -- Using the queries from the <title> fields. Using the V-BM25 weighting model, V-phrase weighting model. No parameters tuning.
2. SRCBEX2 -- Using the queries from the <title> fields. Using the V-DFR_BM25 weighting model, V-phrase weighting model. No parameters tuning.
3. SRCBEX3 -- Using the queries from the <title> fields. Using the V-DFR_BM25 weighting model, V-phrase weighting model. Parameters are tuned using parts of TREC2005 collections and topics.
4. SRCBEX4 -- Using the queries from the <title> fields. Using DFR_BM25 weighting model - whose document length is the profile length – and the V-phrase weighting model. Parameters are tuned by part of TREC 2005 collections and topics.
5. SRCBEX5 -- Using the queries from the <title> fields. Using field-based two-stage search method that uses the V-BM25 weighting model. Parameters are tuned using 8 topics of TREC2006.

The following tables show the evaluation results of the five submitted runs.

Table 3. Results of five submitted runs without support documents

Runs	Average Precision	Bpref	P10
SRCBEX1	0.5290	0.5303	0.6347
SRCBEX2	0.5120	0.5140	0.6204
SRCBEX3	0.5165	0.5172	0.6265
SRCBEX4	0.4793	0.4874	0.5980
SRCBEX5	0.5639	0.5642	0.6551

Table 4. Results of five submitted runs with support documents

Runs	Average Precision	Bpref	P10
SRCBEX1	0.3433	0.4056	0.4694
SRCBEX2	0.3353	0.3989	0.4633
SRCBEX3	0.3384	0.4012	0.4673
SRCBEX4	0.3297	0.3988	0.4653
SRCBEX5	0.3602	0.4299	0.4735

From the above tables, we can see that the average precision of SRCBEX3 is much higher than that of SRCBEX4, which shows that the query-based document length method is more effective. The average precision of SRCBEX2 is almost the same as that of SRCBEX3, which proves that V_DFR-BM25 is stable.

4. Conclusion

We (Ricoh SRCB team) participated in two tasks of Enterprise Track, discussion search and expert search. In the discussion search, we made use of many non-content features, such as timeline and e-mail thread to optimize the search results. We also used an advanced field-based weighting method and query expansion method. Experiments showed that most of these attributes improve effectively the results. In the expert search, we developed a novel two-stage search method and improved it by a field-based approach. Experiments demonstrated these methods are effective.

5. Acknowledgements

The authors would like to thank Ian Soboroff, Nick Craswell, and Arjen P. de Vries for coordinating the Enterprise track. The work is supported by Ricoh Software Research & Development Group, especially by Tetsuya Ikeda, Hideo Itoh and Yinghui Xu. Great thanks to Timothée Bailloeul for checking the manuscript.

References

[1] V.N. Anh, W. Webber, A. Moffat, Melbourne University 2005: Enterprise and Terabyte Tracks. In *Proc. of the Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005. URL: http://trec.nist.gov/pubs/trec14/t14_proceedings.html

[2] N. Craswell, H. Zaragoza, S. Robertson. Microsoft Cambridge at TREC－14: Enterprise track. In *Proc.*

of the Fourteenth Text REtrieval Conference (TREC 2005), Gaithersburg, MD, November 2005. URL: http://trec.nist.gov/pubs/trec14/t14_proceedings.html

[3] O.Vechtomova, M.Kolla, M. Karamuftuoglu. Experiments for HARD and Enterprise Tracks. In *Proc. of the Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005. URL: http://trec.nist.gov/pubs/trec14/t14_proceedings.html.

[4] M. Wu, P. Thomas, D. Hawking. TREC 14 Enterprise Track at CSIRO and ANU. In *Proc. of the Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005. URL: http://trec.nist.gov/pubs/trec14/t14_proceedings.html.

[5] Y. Fu, W. Yu, Y. Li, Y. Liu, M. Zhang, THUIR at TREC 2005: Enterprise Track. Tsinghua University (State Key Lab). In *Proc. of the Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005. URL: http://trec.nist.gov/pubs/trec14/t14_proceedings.html

[6] Y. Cao, H. Li, Microsoft Research Asia, J. Liu, Nankai University, S. Bao, Shanghai Jiaotong University, Research on Expert Search at Enterprise Track of TREC 2005. In *Proc. of the Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005. URL: http://trec.nist.gov/pubs/trec14/t14_proceedings.html.

[7] C. Yao, B. Peng, J. He, Z. Yang, CNDS Expert Finding System for TREC 2005. Peking University, In *Proc. of the Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005. URL: http://trec.nist.gov/pubs/trec14/t14_proceedings.html.

[8] C. Macdonald, B. He, V. Plachouras, I. Ounis, University of Glasgow at TREC 2005: Experiments in Terabyte and Enterprise Tracks with Terrier. University of Glasgow, In *Proc. of the Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005. URL: http://trec.nist.gov/pubs/trec14/t14_proceedings.html.

[9] Z. Ru, Y. Chen, W. Xu, J. Guo, TREC 2005 Enterprise Track Experiments at BUPT. Beijing University of Posts and Telecommunications, In *Proc. of the Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, November 2005. URL: http://trec.nist.gov/pubs/trec14/t14_proceedings.html

[10] S. E. Robertson, S. Walker, M. M. HancockBeaulieu, M. Gatford, and A. Payne. Okapi at TREC-4. In NIST Special Publication 500-236，The Fourth Text REtrieval Conference (TREC-4), pages 73--96, Gaithersburg, MD, 1995.

[11] B. He and I. Ounis. A study of Dirichlet priors for term frequency normalisation. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Pages 465 - 471. Salvador, Brazil. August, 2005.