Structuring and expanding queries in the probabilistic model

OGAWA Yasushi MANO Hiroko NARITA Masumi
HONMA Sakiko
Software Research Center, RICOH Co., Ltd.
1-1-17 Koishikawa, Bunkyo-ku, Tokyo 112-0002, JAPAN
{yogawa,mano,narita,honma}@src.ricoh.co.jp

1 Introduction

This is our first participation in TREC and five runs were submitted for the ad-hoc main task. Our system is based on our Japanese text retrieval system [4], to which English tokenizer/stemmer has been added to process English text. Our indexing system stores term positions, thus providing proximity-based search, in which the user can specify the distance between query terms.

What our system does is outlined as follows:

1. Query construction

   The query constructor accepts each topic, extracts words in each of the appropriate fields and constructs a query to be supplied to the ranking system.

2. Initial retrieval

   The constucted query is fed into the ranking system, which then assigns term weights to query terms, scores each document and turns up a set of top-ranking documents assumed to be relevant to the topic (pseudo-relevant documents).

3. Query expansion

   Based on the feedback from the pseudo-relevant documents, the query expander collects and ranks the words in the pseudo-relevant documents and the words ranked the highest are added to the original query, with the words already in the query re-assigned new term weights.

4. Final retrieval

   The ranking system performs final retrieval using the modified query.

In what follows, we explain what is done in each of the steps in more detail.

2 Query construction

We have employed automatic query processing to construct single-word and phrasal search terms. Our query processing involves a series of steps to identify the important concepts in each topic. In what follows, we describe in some detail how single and phrasal search terms are created by linguistic methods and represented as a structured query.

2.1 Stemming and morphological expansion of query terms

Natural language text in each topic is processed by our English tokenizer and stemmer to output stemmed word tokens. We have developed a new stemmer which conflates morphologically related words in two steps: the first step stems terms for document indexing and query processing and the second performs morphological expansion of query terms. To avoid degradation of performance by overstemming, our stemmer only stems an inflected form to its base form and a sequence of derivational suffixes to the initial suffix. [1] For example, "humanization" and "humanized" are stemmed to "humanize," but not "humanize" to "human." When constructing queries, each search term is stemmed and expanded with its morphological variants; "humanize" is possibly expanded with "human," "humanity," "humanise," and so on.

Although a run with stemmed queries without morphological expansion showed rather consistent improvements over a run without stemming, expansion of query terms produced inconsistent results. Some queries benefited a lot; others were damaged a lot. We found that both benefits and damages resulted from derivational variants. To the contrary, spelling variants rarely had ill effects although they improved only a few queries. Avoiding risks, we decided to expand terms only with spelling variants for our submitted runs. Thus, a stemmed term "humanize" is expanded to #SYN(humanize,humanise) with a synonym operator #SYN.

2.2 Single term selection

From the terms extracted by the stemmer, the query constructor selects single-word search terms for the query by eliminating very common and irrelevant words, i.e., stopwords. We have used two kinds of stopword lists, the Fox's [1] word list for the <title> field and its augmented word list we created for the <desc> field. To augment the Fox's word list, some dozens of unimportant words were manually added to the original list after examination of the <desc> fields from TREC-3 to TREC-7.

For example, the words "identify," "document," and "discuss" in "Identify documents that discuss clothing sweatshops" were added to the Fox's word list because these words provide no information about the information need.

2.3 Phrasal term selection

Syntactic phrases are recognized in the natural language text by applying the syntactic chunker LT_CHUNK developed at the Edinburgh Language Technology Group. This chunker uses the part-of-speech information provided by the tagger LT_POS and identifies boundaries of simple noun phrases which do not include prepositional or clausal post-modifiers.

Each noun phrase is tokenized/stemmed and then stripped of all stopwords. As a result, the phrases consisting of two or more single words are extracted for use in search terms.

For phrases consisting of three or more single words, we have given a special treatment because we have experimentally found out that these multi-word phrases are less likely to match documents in the collection. First, all pairs of single words are derived from the target phrase. For example, the noun phrase "industrial waste disposal" is decomposed to derive three possible word pairs such as "industrial waste," "waste disposal," and "industrial disposal." Second, the word pairs which never occur in sequence in the TREC test collections are discarded. In the above example, the last word pair "industrial disposal" is discarded by this processing.

[1] Our decision to what extent we should stem a word is partly based on [3] and [2].

Further, we handle hyphenated words as phrasal terms by specifying that the constituent words be found adjacent in a document.

2.4 Query representation

Single and phrasal search terms are combined into a query using syntax of our query language. As mentioned above, term variants by morphological expansion are expressed with a synonym operator #SYN. Phrasal terms are treated as arguments to a proximity operator #WINDOW and two forms of representation are provided with different window sizes and constraints on word order.

We have also introduced a scoring operator #SCALE to phrasal term representation to adjust its term weight because our preliminary experiments suggest that a phrasal term should be given a lower term weight than a single term. After a series of experiments on weight adjustment, we have fixed two different combinations of weight scales, respectively, for phrasal terms from the <title> field only and those from the <title> and <desc> fields.

To sum, our sample queries are expressed as follows:

```
A query from the <title> field only:
     #OR(industrial,waste,disposal,
         #SCALE[0.1](#WINDOW[1,1,o](industrial,waste)),
         #SCALE[0](#WINDOW[2,500,u](industrial,waste)))

A query from the <title> and <desc> fields:
     #OR(killer,bee,attack,human,#SYN(africanize,africanise),
         #SCALE[0.4](#WINDOW[1,1,o](killer,bee)),
         #SCALE[0.25](#WINDOW[2,500,u](killer,bee)))
```

where #WINDOW[1,1,o] specifies that the two words be found adjacent in a document while #WINDOW[2,500,u] specifies that the two words not be found adjacent but occur within a window of 500 words with no constraint on word order. Note also that single terms are merged with phrasal terms using a logical operator #OR.

3 Initial retrieval

For each query constructed by the query constructor, the ranking system ranks the documents in the target document collection and retrieves top-ranking documents. To rank documents, the system uses term weighting and document scoring formulae similar to Okapi's but with some modifications, mostly in term weighting.

In the probabilistic model [5], each term in the query is assigned a term weight to represent the appropriateness of the term as a discriminator in the collection. Terms are weighted according to

$$w_t = \log \frac{p}{1-p} - \log \frac{q}{1-q} \tag{1}$$

where p is the probability that a document contains the term, given that it is relevant and q is the probability that a document contains the term, given that it is not relevant.

In Okapi [6], the probabilities p and q are given by

$$p = \frac{p_0}{p_0 + (1-p_0)\frac{N-n}{N}} \tag{2}$$

$$q = \frac{n}{N} \tag{3}$$

where N is the number of documents in the collection, n is the number of the documents in which the term occurs and p_0 is the estimate of the probability that the term occurs in a relevant document when no document contains the term. From (1), (2) and (3), we have

$$w_t = \log \frac{p_0}{1-p_0} + \log \frac{N}{N-n} - \log \frac{n}{N-n}. \tag{4}$$

By replacing $\log \frac{p0}{1-p0}$ with k_4, we have

$$w_t = k_4 + \log \frac{N}{n}$$

where $-\infty < k_4 < \infty$ since $0 \leq p_0 \leq 1$.

Now, with this weighting formula, k_4 is less than zero when p_0 is estimated as smaller than 0.5, which is usually a reasonable estimate. However, as has been pointed out, with the value of k_4 negative, the term weight could result in a negative value depending on the value of n, the consequence of which would be degenerate retrieval.

To solve this problem, we have changed the way how the probability p is estimated. That is, in our modified formula, p is estimated as

$$p = p_0 + (1-p_0)\frac{n}{N}. \tag{5}$$

From (1), (5) and (3), we have

$$w_t = \log \left(\frac{p_0}{1-p_0} \cdot \frac{N}{N-n} + \frac{n}{N-n} \right) - \log \frac{n}{N-n} \tag{6}$$

and if we let k_4' be $\frac{p_0}{1-p_0}$, we have

$$w_t = \log \left(k_4' \cdot \frac{N}{n} + 1 \right).$$

Note that with our formula, the value of k_4' never gets negative regardless of the value of p_0, thus ensuring that the term weights are always positive. By keeping the term weights positive, the quality of retrieval is maintained even in the worst case.

With each term weighted according to the above formula, the ranking score for each document is calculated using a formula very similar to Okapi's.

$$s_{d,q} = \sum_{t \in q} \frac{w_t}{k_4 + \log N} \cdot \frac{f_{t,d}}{k_1((1-b) + b\frac{d_t}{d_{ave}}) + f_{t,d}}$$

where $f_{t,d}$ is the within-document frequency of the term, d_t is the document length d_{ave} is the average document length, k_1 and b are parameters, just as in Okapi's.

4 Query expansion

After initial retrieval, the query expander collects single terms in the pseudo-relevant documents and ranks them according to its Term Selection Value (TSV) while reweighting query terms, using formulae adopted from Okapi's and modified in three ways as described below. The top-ranking single terms are then added to the original query with their respective term weights. The single terms and phrasal terms origianlly included in the query are also given re-assigned term weights, multiplied with a bonus factor.

4.1 Term weighting 1

The first of the modifications in the TSV formula involves a change in term weighting after initial retrieval, from Okapi's to the one that reflects the modification in term weighting during initial retrieval mentioned above.

In Okapi, term weights are re-assigned after initial retrieval, when feedback from retrieved documents becomes available. The new term weights are calculated as a weighted average of the term weight estimated without any relevance feedback and the term weight estimated solely from relevance feedback. That is, if we put the term weight estimated without feedback, which was assigned for initial retrieval, as

$$w_t = \log \frac{p}{1-p} - \log \frac{q}{1-q} = w_p - w_q,$$

the new term weight after relevance feedback would be

$$w_t = C_p * w_p + (1 - C_p) * w'_p - C_q * w_q - (1 - C_q) * w'_q \tag{7}$$

where w'_p and w'_q are term weight components based on relevance feedback and C_p and C_q are coefficients. Specifically, with

$$w_p = \log \frac{p_0}{1-p_0} + \log \frac{N}{N-n} = k_4 + \log \frac{N}{N-n}, \ w_q = \log \frac{n}{N-n} \tag{8}$$

from (4), the term weight after feedback in Okapi is calculated as

$$\begin{aligned} w_t \quad = \quad & \frac{k_5}{k_5+\sqrt{R}}(k_4 + \log \frac{N}{N-n}) + \frac{\sqrt{R}}{k_5+\sqrt{R}} \log \frac{r+0.5}{R-r+0.5} \\ & - \frac{k_6}{k_6+\sqrt{S}} \log \frac{n}{N-n} - \frac{\sqrt{S}}{k_6+\sqrt{S}} \log \frac{s+0.5}{S-s+0.5} \end{aligned}$$

where R is the number of relevant documents, r is the number of relevant documents containing the term, S is the number of non-relevant documents, s is the number of non-relevant documents containing the term, and k_5 and k_6 are parameters.

In our system, we followed the same principle as Okapi's, adopting (7). However, as described earlier, since we changed initial w_p from (8) to

$$w_p = \log \left(\frac{p_0}{1-p_0} \cdot \frac{N}{N-n} + \frac{n}{N-n} \right) = \log \left(k'_4 \cdot \frac{N}{N-n} + \frac{n}{N-n} \right)$$

as shown in (6), term weighting after feedback was changed accordingly, resulting in the formula

$$\begin{aligned} w_t \quad = \quad & \frac{k_5}{k_5+\sqrt{R}} \log \left(k'_4 \frac{N}{N-n} + \frac{n}{N-n} \right) + \frac{\sqrt{R}}{k_5+\sqrt{R}} \log \frac{r+0.5}{R-r+0.5} \\ & - \frac{k_6}{k_6+\sqrt{S}} \log \frac{n}{N-n} - \frac{\sqrt{S}}{k_6+\sqrt{S}} \log \frac{s+0.5}{S-s+0.5}. \end{aligned}$$

4.2 Term weighting 2

The above term weighting formula underwent further modification when we noticed some incidents of words that are too common to be useful in a query appearing in the expansion terms selected using the formula. To keep these words from being included in expansion, the term weighting formula was changed to reflect the document frequency of the term in the coefficient part of the formula as

$$\begin{aligned} w_t \quad = \quad & \frac{k_5}{k_5 + \sqrt{\frac{R}{R+n-r}}} \log\left(k_4' \frac{N}{N-n} + \frac{n}{N-n}\right) + \frac{\sqrt{\frac{R}{R+n-r}}}{k_5 + \sqrt{\frac{R}{R+n-r}}} \log \frac{r+0.5}{R-r+0.5} \\ & - \frac{k_6}{k_6 + \sqrt{\frac{S}{S+n-s}}} \log \frac{n}{N-n} - \frac{\sqrt{\frac{S}{S+n-s}}}{k_6 + \sqrt{\frac{S}{S+n-s}}} \log \frac{s+0.5}{S-s+0.5}, \end{aligned}$$

thus reducing the relative weight of terms appearing in both the pseudo-relevant documents and the whole document collection. (In the experiments, S was set to 0.)

4.3 Term Selection Value

We also looked at the whole TSV formula of Okapi's [7]

$$TSV = (r/R - \alpha \cdot s/S) \cdot w_t$$

where α is a parameter. From our observation on the terms selected with this formula, however, we felt that it would be better to eliminate those terms too specific to serve as expansion terms, such as a telephone number, that were included in the selection. To that end, the Okapi's TSV formula was modified to reflect the within-document frequency as

$$TSV = \left(\sum_{d \in R} \frac{f_{t,d}}{k_1((1-b) + b\frac{d_t}{d_{ave}}) + f_{t,d}} \Big/ R - \beta \cdot \sum_{d \in S} \frac{f_{t,d}}{k_1((1-b) + b\frac{d_t}{d_{ave}}) + f_{t,d}} \Big/ S \right) \cdot w_t$$

where β is a parameter, which would lower the TSV of those terms that occur only once or twice in a document, in comparison with the terms that occur more often per document. (We also requred r to be larger than 1 for a term to be considered. Also, S was set to 0 in the experiment as in term weighting.)

With these modifications, our expansion method is more likely to select terms neither too common throughout the collection nor too rare to be appicable to the whole collection, without resorting to such cut-off measures as a stopword list or a filter to exclude, for instance, words containing digits.

5 Final retrieval

The expanded-and-reweighted query is sent to the ranking system and documents are retrieved as final result. Document ranking is done just as in initial retrieval, except that the term weights are supplied by the query.

6 Results

We produced eight runs using different combinations of the following conditions of the queries:

- Queries using only <title> and queries using <title> and <desc>

- Queries using no phrasal search terms and queries using phrasal search terms
- Queries using no expansion and queries using expansion

We tuned up the formulae using mainly queries generated from the TREC-7 topics. For runs with expansion, title-only queries were mostly used for the tuneup. Parameters we chose for each of the eight runs are listed in Table 1 – 3. Note that in Table 1 and Table 2, we chose different sets of parameter values for the same retrieval parameters. This is because, for retrieval in a run with no query expansion, we wanted parameter values that would maximize average precision, whereas for initial retrieval for a run with query expansion, we looked for parameter values that would maximize precision at ten retrieved documents.

Table 1: Parameters for runs without expansion

	no phrases		phrases	
	title only	title+desc	title only	title+desc
k_1	0.75	1.00	0.50	0.75
b	0.25	0.25	0.25	0.25
k'_4	0.20	0.20	0.05	0.05
Scale for #WINDOW[1,1,o]	–	–	0.10	0.40
Scale for #WINDOW[2,500,u]	–	–	0.00	0.25

Table 2: Parameters for runs with expansion (initial retrieval)

	no phrases		phrases	
	title only	title+desc	title only	title+desc
k_1	0.50	1.00	0.75	1.00
b	0.25	0.25	0.25	0.25
k'_4	0.50	0.10	0.20	0.20
Scale for #WINDOW[1,1,o]	–	–	0.10	0.25
Scale for #WINDOW[2,500,u]	–	–	0.10	0.10

Table 3: Parameters for runs with expansion (final retrieval)

	no phrases		phrases	
	title only	title+desc	title only	title+desc
Number of documents used for expansion	10	10	10	10
k_1	0.75	1.00	0.75	0.75
b	0.25	0.25	0.25	0.25
k'_4	0.20	0.20	0.10	0.05
k_5	0.25	0.25	0.25	0.25
Scale for #WINDOW[1,1,o]	–	–	0.40	0.40
Scale for #WINDOW[2,500,u]	–	–	0.25	0.25
Maximum number of terms to be added	25	30	25	30
Minimum number of terms to be added	10	10	10	10
Minimum r for term to qualify	2	2	2	2
Bonus factor for query terms	3.5	4.0	3.5	4.0

The experimental runs using the above parameters resulted in the following average precision measurements (Table 4). The table clearly shows improvement in performance when phrasal search terms are used and when queries are expanded, especially for queries using both <title> and <desc>.

Table 4: Average precision for TREC-7 topics

	title only	title + desc
no phrases/no expansion	0.2033	0.2127
phrases/no expansion	0.2120	0.2373
no phrases/expansion	0.2513	0.2571
phrases/expansion	0.2584	0.2838

Using the same parameters as above, the results for the TREC-8 topics are shown in Table 5.

Table 5: Average precision for TREC-8 topics

	title only	title + desc
no phrases/no expansion	0.2560	0.2363
phrases/no expansion	0.2572	0.2633
no phrases/expansion	0.2647	0.2426
phrases/expansion	0.2689	0.2748

In TREC-8 runs, the effect of expansion was not as great as in TREC-7 runs as far as the above results are concerned. This may be because the parameter values we experimented with were not optimal. In fact, by re-adjusting the parameters, especially the bonus factor for query terms, we saw improvement in the average precision – for example, 0.2800 for title-only queries with no phrases.

References

[1] C. Fox. A stop list for general text. *ACM SIGIR Forum*, Vol. 24, No. 2, pp. 19–35, 1991.

[2] D.A. Hull. Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, Vol. 47, No. 1, pp. 70–84, 1996.

[3] R. Krovetz. Viewing morphology as an inference process. In *Proc. of 16th ACM SIGIR Conf.*, pp. 191–203, 1993.

[4] Y. Ogawa and T. Matsuda. An efficient document retrieval method using n-gram indexing (in Japanese). *Transactions of IEICE*, Vol. J82-D-I, No. 1, pp. 121–129, 1999.

[5] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Jounal of the American Society for Information Science*, Vol. 27, pp. 129–146, 1976.

[6] S.E. Robertson and S. Walker. On relevance weights with little relevance information. In *Proc. of 20th ACM SIGIR Conf.*, pp. 16–24, 1997.

[7] S. Walker et al. Okapi at trec-6: Automated ad hoc, vlc, routing, filtering and qsdr. In *Proc. of 6th Text REtrieval Conf.* NIST, 1996.