Experiments on Proximity Based Chinese Text Retrieval in TREC 6

K. Rajaraman[1], Kok F. Lai and Y. Changwen
Information Technology Institute
11, Science Park Road, Singapore 117685.

Abstract

In TREC 6, we participate in the Chinese track and report our experiments on proximity based text retrieval. Our participation this year concentrates on automatic retrieval methods natural for the Chinese language.

We index the documents by treating every Chinese character as a single term and store positional information for all terms. During retrieval we employ a proximity operator that uses the positional information in the index, to rank the documents. The operator is defined such that documents are scored in proportion to the proximity of characters as they appear in the query. Since we only use the proximity of characters to compute the score, the algorithm does not strictly require the word boundaries be known a priori. In particular, phrase detection can be derived as a special case of our algorithm by giving maximum score when the characters are immediately adjacent and 0 otherwise. This indexing and retrieval scheme is significantly different from our TREC 5 method.

We submit three official runs itich1, itich2 and itich3 for TREC 6. For itich3, we use all phrases from the Description field and compute scores with our proximity operator. The runs itich1 and itich2 are obtained through automatic query expansion methods. We dynamically build a 3-gram phrase dictionary from top 20 documents for each query ranked in itich3 and pick phrases to expand from this dictionary using document frequency estimates. The run itich2 is different from itich1 in that the expanded phrases are filtered to remove duplicate and common phrases.

1 Introduction

In TREC 5, we participated in Routing, Filtering and Chinese tracks[NL96]. Due to time and space constraints, this year we take part in the Chinese track only. Our last year experiments include both automatic and manual Chinese text retrieval. Our experiments in TREC 6 concentrate on automatic methods natural for the Chinese language.

It is well known that Chinese text is written as a string of ideograms with no specific word delimitter as in languages like English. Words are identified based on the context in

[1]Please direct all enquiries and correspondence to K. Rajaraman, Information Technology Institute, 11, Science Park Road, Singapore 117685. Email : kanagasa@iti.gov.sg

which they appear. This distinguishing feature of Chinese language (and a few other Asian languages) demands a departure from conventional methods for Chinese IR.

Broadly there are two approaches to Chinese text retrieval *viz.* linguistic and non-linguistic. Of these, the non-linguistic approach is the best investigated one for large applications[BSM96, ACC+96, KG96, BGH+96, GCH+96]. This approach is based on representing Chinese in an English like structure and applying conventional IR techniques for English. Typically, a word segmenter (eg. longest match algorithm. See, for instance, [BGH+96]) is applied on the text collection and the segmented words used to build the index. During retrieval, the query is segmented in a similar way and the resulting words matched with the index to score the documents. Stopword removal and weighting schemes like *tf.idf* are employed to improve precision and recall. As it can be observed, this is essentially a process of mimicing English IR for Chinese by using segmented words as index terms. This method crucially depends on the accuracy of the segmentation algorithm. For instance, if the segmentation algorithm returns the longest word but the query contains only a short word, then the result may be a partial match. This is an inherent problem with word-based indexing. Character based indexing is a more flexible method. In character based indexing, either 1-grams, 2-grams or 3-grams can be used as index terms. 1-gram terms are ambiguous and they may adversely affect precision. 2-grams and 3-grams carry more specific meaning than 1-grams and are usually adequate as index terms[LA96, Lin72]. However, the size of the 2-grams index may be too large to be manageable. Even for a collection with 7000 characters (in GB set), theoretically there could be 49 million 2-gram terms. Hence, to tackle the retrieval problem effectively, we need a novel approach more natural for the Chinese language. Our TREC 6 work is a step in this direction.

2 Proximity Based Text Retrieval

Our approach is based on ranking documents using the proximity of characters in the query string.

We index every character in the document collection and store the positional information of all occurences of the terms. The positional information will be used in scoring the documents as below.

Suppose $c_1c_2....c_n$ is a Chinese string. We define a proximity operator

$$Prox_i(c_1c_2...c_n) = \sum_{k=1}^{n-1} \frac{f\left(Dist_i(c_k, c_{k+1})\right)}{n-1}$$

where i stands for the document number and

$$Dist_i(c_k, c_{k+1}) = \text{smallest positive distance from } c_k \text{ to } c_{k+1} \text{ in } i\text{-th document}$$

i.e. If we define

$$pos_{ij}(c_k) = \begin{cases} \text{the position of } j\text{-th occurrence} & \text{if } c_k \text{ occurs at least } j \text{ times} \\ \text{of character } c_k \text{ in } i\text{-th document,} & \text{in the } i\text{-th document} \\ 0, & \text{otherwise} \end{cases}$$

then

$$Dist_i(c_k, c_{k+1}) = min_{j,l}(max(0, pos_{ij}(c_{k+1}) - pos_{il}(c_k)))$$

The function $f : R \rightarrow [0, 1]$ (called the *proximity* function), is non-decreasing on $(-\infty, 0)$ and non-increasing on $[0, \infty)$. In this paper we use the following proximity function.

$$f(x) = \begin{cases} \frac{1}{x^2}, & x < C \\ 0, & \text{otherwise} \end{cases}$$

where the constant C is chosen to be small enough to avoid undesirably long matches during proximity computation.

By definition, given a Chinese query, the operator computes a [0,1] score for every document. Using the positional positional information in the index, the score is made proportional to how proximal the query string characters are in the document under consideration. (Hence the name "proximity" operator.)

It can be noted that this scoring method automatically takes cares of the word boundaries. This follows because if the word boundary were at c_k, then c_k and c_{k+1} will not be adjacent (except under pathological cases) in the document and $Dist_i(c_k, c_{k+1})$ would be zero by definition. Hence the operator would score every document as though the query were properly segmented. An important consequence of this observation is that word segmentation is *not* needed for our method to work. We believe that this method is more natural for Chinese IR than the conventional techniques used by majority of the TREC systems.

We next describe our experiments for TREC 6.

3 Our Experiments

For our first run, we used phrases from the description field and ranked the documents using the proximity based scoring method described above.

Results:

```
Total number of documents over all queries
     Retrieved:    26000
      Relevant:     2958
       Rel_ret:     2215
Interpolated Recall - Precision Averages:
    at 0.00       0.8100
    at 0.10       0.6137
    at 0.20       0.5222
    at 0.30       0.4842
    at 0.40       0.3930
    at 0.50       0.3446
    at 0.60       0.3047
    at 0.70       0.2362
    at 0.80       0.1777
    at 0.90       0.1148
    at 1.00       0.0252
Average precision (non-interpolated) over all rel docs
                  0.3427

Precision:
  At    5 docs:   0.6385
  At   10 docs:   0.6038
  At   15 docs:   0.5821
  At   20 docs:   0.5692
  At   30 docs:   0.5051
  At  100 docs:   0.3715
  At  200 docs:   0.2671
  At  500 docs:   0.1501
  At 1000 docs:   0.0852
R-Precision (precision after R (= num_rel for a query) docs retrieved):
       Exact:     0.3881
```

This is our baseline experiment. (In comparison with our TREC-5 automatic retrieval method[NL96], the new algorithm is observed to show a over 15% improvement in average precision.) This run is called itich3.

For the next experiment, we extracted the top 20 ranked documents for each query from the above and built a 3-gram dictionary from these documents. Rare and common 3-grams were filtered based on term occurrence estimates while building the dictionary. We ranked the entries in the dictionary according to the measure

$$\frac{\text{number of documents containing the phrase}}{\log(\text{number of documents containing the phrase in the whole collection})}$$

and picked top 10 phrases from this dictionary. These phrases are used to expand the query. We assigned a weight of 0.5 for the expanded phrases while the phrases in the original query carried full weight. This run is called itich1.

Results:

```
Total number of documents over all queries
      Retrieved:    26000
       Relevant:     2958
        Rel_ret:     2447
Interpolated Recall - Precision Averages:
    at 0.00       0.8910
    at 0.10       0.7352
    at 0.20       0.6477
    at 0.30       0.6092
    at 0.40       0.5523
    at 0.50       0.4936
    at 0.60       0.4323
    at 0.70       0.3437
    at 0.80       0.2551
    at 0.90       0.1688
    at 1.00       0.0423
Average precision (non-interpolated) over all rel docs
                  0.4541

Precision:
  At    5 docs:   0.7385
  At   10 docs:   0.7192
  At   15 docs:   0.7026
  At   20 docs:   0.6577
  At   30 docs:   0.6115
  At  100 docs:   0.4615
  At  200 docs:   0.3158
  At  500 docs:   0.1644
  At 1000 docs:   0.0941
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.4755
```

The results show that query expansion improves the average precision by over 32%. Appreciable improvement in recall is also observed.

For our last experiment, we expanded phrases as above but filtered out duplicate and common phrases (like dates). This run was called itich2.

Results

```
Total number of documents over all queries
     Retrieved:    26000
      Relevant:     2958
       Rel_ret:     2349
Interpolated Recall - Precision Averages:
    at 0.00       0.8698
    at 0.10       0.7248
    at 0.20       0.6206
    at 0.30       0.5649
    at 0.40       0.4757
    at 0.50       0.4264
    at 0.60       0.3709
    at 0.70       0.3005
    at 0.80       0.2110
    at 0.90       0.1321
    at 1.00       0.0155
Average precision (non-interpolated) over all rel docs
                  0.4145

Precision:
  At    5 docs:   0.7538
  At   10 docs:   0.7192
  At   15 docs:   0.6872
  At   20 docs:   0.6500
  At   30 docs:   0.5885
  At  100 docs:   0.4288
  At  200 docs:   0.2973
  At  500 docs:   0.1590
  At 1000 docs:   0.0903
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.4452
```

The above run, though better than itich1, is actually worse compared to itich1.

An analysis of the performance of three submitted runs is shown below:

itich1 is our best run followed by itich2. We still have no convincing explanation as to why itich2 is poorer than itich1, but we believe it could be due to term weighting. We found that the conventional *tf.idf* weighting coupled with the proximity operator hurts performance. (With just idf weight, however, the performance improves but not significantly on average.) Hence our offical runs included only un-weighted retrieval results. The effect can be seen from Table 3.1. While the Recall@1000 was at or above median for 14 queries for itich1, the avg. precision was above median on only 6 queries. We feel a novel weighting

Run	Average	At or above median on		
	Precision	Avg Precision	Recall@100	Recall@1000
itich1	0.4541	6 queries	11 queries	14 queries
itich2	0.4145	4 queries	7 queries	10 queries
itich3	0.3427	4 queries	6 queries	8 queries

Table 3.1: Performance of submitted runs

scheme needs to be devised to work with the proximity operator. We plan to investigate this as part of our future work. Our future investigations will also include more sophisticated proximity operators.

References

[ACC+96] James Allan, Jamie Callan, Bruce Croft, Lisa Ballesteros, John Broglio, Jinxi Xu, and Hongmin Shu. INQUERY at TREC-5. In *Text REtrieval Conference(TREC-5)*. NIST, Gaithersburg, Maryland, 1996.

[BGH+96] M.M. Beaulieu, M. Gatford, Xiangxi Huang, S.E. Robertson, S. Walker, and P. Williams. Okapi at TREC-5. In *Text REtrieval Conference(TREC-5)*. NIST, Gaithersburg, Maryland, 1996.

[BSM96] Chris Buckley, Amit Singhal, and Mandar Mitra. Using query zoning and correlation with SMART: TREC 5. In *Text REtrieval Conference(TREC-5)*. NIST, Gaithersburg, Maryland, 1996.

[GCH+96] Fredric C. Gey, Aitao Chen, Jianzhang He, Liangjie Xu, and Jason Meggs. Term importance, boolean conjunct training, negative terms and foreign language retrieval. In *Text REtrieval Conference(TREC-5)*. NIST, Gaithersburg, Maryland, 1996.

[KG96] K.L. Kwok and L. Grunfeld. TREC-5 English and Chinese retrieval experiments using PIRCS. In *Text REtrieval Conference(TREC-5)*. NIST, Gaithersburg, Maryland, 1996.

[LA96] Joon Ho Lee and Jeong Soo Ahn. Using n-grams for Korean text retrieval. In *Proc. of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 216–224. ACM, New York, 1996.

[Lin72] Y.T. Lin. *Chinese English Dictionary of Modern Usage.* Chinese University of Hong Kong Press, Hong Kong, 1972.

[NL96] Chong-Wah Ngo and Kok F. Lai. Experiments on Routing, Filtering and Chinese text retrieval in TREC-5. In *Text REtrieval Conference(TREC-5).* NIST, Gaithersburg, Maryland, 1996.