# PSM: A new Re-ranking algorithm for Named-page

Jiafeng Guo, Lin Ding, Gang Zhang, Yue Liu, Xueqi Cheng
Intelligent Software Department, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
guojiafeng@software.ict.ac.cn

## 1. Introduction

This year, the IR group of ICT participated in the terabyte track named-page Finding subtask for the first time. Since the document collection is as large as about 426G, our most important goal is to find an efficient way to catch the target web page in such a huge size data set. Meanwhile we want to make the indexing and retrieval processing at a reasonable low cost, both on hardware and time-consuming. We used our "FirteX" engine for indexing and retrieval of this task. The indexing time is within 15 hours and the retrieval time is short enough(less than 2 seconds per query). The main contribution of our work is that we design a Pattern Similarity Matching(PSM) re-ranking algorithm to reorder the results and rank the target document as top 1 as possible. We were glad to see that we've got an exciting performance on the last year's (2005) topics during our experiment. The chief procedure of our work can be divided into three parts as below, which are data preprocess, indexing and retrieval, and re-ranking.

## 2. Data preprocess

At the first beginning, we need to preprocess the data, which means to extract text from the html document of the .gov2 collection and filter all the tags and scripts. At the same time, we checked the document type to determine whether it is a html document or a PDF document or anything else. We threw away the anchor text and reserved some structural information of the html document for later use. We ignored the anchor text because we didn't do any link analysis in our system. The structural information we reserved here are the title and url fields.Title, as a general or descriptive heading of a document, has been proved to be a high-quality information source in past experiments. Here, we extract title between the <TITLE> and </TITLE> tags. And we found that about 6% pages in .gov2 collection have absent title field or default title such as "untitlted".URL is also important in this task. Since we are required to find a particular page, some key words of the query may appear in the URL of such pages as we analyze the last year's named-page finding results. If we can recognize it, this information may help us a lot to locate these pages.

After this preprocessing, the data size is about 152G, only 1/3 of the original data set. Data preprocessing phase used 454 minutes.

## 3. Indexing and retrieval

Indexing large data set as fast as possible is one of the most important targets of our system. We made this possible by fast parsing module, efficient memory management, and powerful index compressing. All the indexing time(including parsing procedure) is about 229 minutes, which means we've got an index-building speed of about 10M/s.The size of full text index is about 46G.

In retrieval procedure, performance is also considered carefully, both on MRR and

time-consuming. A modified BM25 formula [1][2] was used to calculate the similarity between the document and query. As we indexed title, main body and URL, we give different priorities to different fields so that to make the text from the title and URL more important. Some previous work has investigated the importance of combining the Structural Information of these fields[3].Here we modified the value of the tf to show this priority difference and combine them together. We call this new term frequency value a weighted term frequency. The formulas we used are expressed as below:

$$bm25(q,d) = \sum_{t \in q} \log\left(\frac{N - f_t + 0.5}{f_t + 0.5}\right) \times \frac{(k1+1) \times wf_{d,t}}{K + wf_{d,t}} \qquad (1)$$

Where:

$t$ = query terms

N = Number of documents in the collection

$f_t$ = Number of documents that a term $t$ occurs in

and

$$\mathrm{w}f_{\mathrm{d},t} = f_{\mathrm{d},t} + \alpha * \log\left(AF \big/ f_t + 1\right) \qquad (2)$$

Where:

$f_{\mathrm{d},t}$ = Number of times that a term $t$ occurs in document $d$

AF = Average $f_t$ over the collection

$\alpha$ is the parameter

and

$$\mathrm{K} = k_1\left((1-b) + b \times L_d \big/ AL\right) \qquad (3)$$

Where:

$L_d$ = Length of document d in bytes,

AL = Average document length over the collection

$k_1$ and b are parameters

We use the default parameter settings for the basic BM25 parameters. The parameters which we added use a heuristic value.

## 4. Re-ranking

The Named-page task requires us to return the target document No.1 rank as possible. But if we only use our modified BM25 formula to score and rank the documents , the result is a little bit disappointed. Some target documents are not in the top 1000 rank. The main reason may be caused by the essence of the BM25 formula. It treats the document and query as a bag of words. This may encounter much trouble when many documents contain these query words but they are not related. So beyond the traditional retrieval method and similarity scoring, we have tried a new re-ranking algorithm called Patten Similarity Matching (PSM) to re-organize the results we've got from the retrieval procedure. This re-ranking method is different from BM25 as it treats the

document and query as a sequence of words. By this way, it can use some algorithm like pattern match to show some hidden similarity properties between the document and query we haven't revealed before.

First, we suppose that during the retrieval procedure, we can obtain the target document in the returned result set. Then, we apply our re-ranking algorithm PSM to this set to get a new score for each result. After that, we use a quick partial sort algorithm to re-rank the result set and return as our final results.

The PSM algorithm chiefly considered the meaning of the query by the way of the words' order and position. We have observed that many documents, which contain some/all the words of the query, may talk about something not related to the topic at all. The key differences between these documents and target document are those words' order and position. If we changed the words order and position, we changed the meaning. Here are two observations we've got:

1.If the words' order in the document is more like the query , the document is more relative

2.If the words' distance in the document is more close to each other , the document is more relative

Upon these observations, we can easily get the goal of our PSM algorithm as follows:

1.If there exits an exact pattern of the sequence of query words in the document, then it get a score of 1

2.If there exits absolutely no such pattern of the sequence of query words in the document, then it get a score of 0

3.If the document misses some words of the query, give some punishment to it

Therefore, we can define the similarity function between a pattern P and a Query Pattern Q like this:

$$Sim(P,Q) = \begin{cases} 0 & \text{if P is a Blank Pattern} \\ \dfrac{1}{n-1} \displaystyle\sum_{k_1k_2 \in sub(Q)} \dfrac{1}{dist_P(k_1,k_2)} & \text{otherwise} \end{cases}$$

Where

P is a Blank Pattern means no word in Query Q appears in P

n = number of key words of the Q

$k_1$,$k_2$ are two key words of Q and $k_1k_2$ is substring of Q

If all the key words in the Query Pattern appear in this pattern, the formula is uniform:

$$dist_p(k_1,k_2) = pos(k_2) - pos(k_1)$$

Where

$pos(k_i)$ is the absolute position of the word $k_i$ in the document

However, if some of the key words in the Query Pattern are missing here, the formula has a little change:

$$dist_P(k_1,k_2) = \begin{cases} punishment & k_1 \text{ or } k_2 \text{ is missing or both are missing} \\ pos(k_2) - pos(k_1) & \text{otherwise} \end{cases}$$

punishment we used in experiment is defined as the length of the document. Thus, the similarity between the document and the query can be expressed like this:

$$Sim_{PSM}(D,Q) = \max_{P \in pat(D)} Sim(P,Q)$$

Where
pat(D) is the set of all the possible patterns that document D contains.

So as we can see , PSM algorithm checked each result document and calculate the score in a reasonable way, referenced to the words' order and position in the query. We made some experiments on the 2005 named-page topics. It is not difficult to find that it is not enough to only use the PSM to re-rank the results list. The reason is that it is from a new way to reveal the similarity between the query and document and it loses the information which BM25 carries. Therefore, we considered to use the combination of the PSM similarity score with the BM25 score to re-rank the results list. Here we use a linear formula to combine the two similarity scores and it is shown in the following:

$$Sim(D,Q) = \begin{cases} Sim_{BM25}(D,Q) + \beta * Sim_{PSM}(D,Q) & if\ Sim_{PSM}(D,Q) > threshold \\ Sim_{BM25}(D,Q) & otherwise \end{cases}$$

and we've found significant improvement of the MRR compared with results which used BM25 only.

| **Run tag** | **MRR** | **Succ in top 10** | **Failure in top 1000** | **Use of Anchor?** | **Use of Structure?** |
|---|---|---|---|---|---|
| BM25 only | 0.428 | 57.1% | 18.3% | N | Y |
| PSM only | 0.233 | 35.4% | 34.9% | N | Y |
| Combined | 0.472 | 63.5% | 15.9% | N | Y |

## 5. Environment and Official Runs

The Hardware and software we used to experiment is showed in the table below.

| | **CPU** | **RAM** | **Hard disk** | **OS** | **Compiler** |
|---|---|---|---|---|---|
| PC | AMD Opteron 848 | 8G | 320G | Linux | GCC |

Here, we give the description and the results of the 5 official runs submitted, which are listed in the table below.

| **Run tag** | **MRR** | **Succ in top 1** | **Succ in top 5** | **Succ in top 10** | **Failure in top 1000** |
|---|---|---|---|---|---|
| **icttb0600** | 0.332 | 48 (26.5%) | 75 (41.4%) | 81 (44.8%) | 52 (28.7%) |
| **icttb0601** | 0.332 | 48 (26.5%) | 75 (41.4%) | 81 (44.8%) | 52 (28.7%) |
| **icttb0602** | 0.334 | 49 (27.1%) | 75 (41.4%) | 81 (44.8%) | 53 (29.3%) |
| **icttb0603** | 0.337 | 49 (27.1%) | 75 (41.4%) | 80 (44.2%) | 52 (28.7%) |
| **icttb0604** | 0.330 | 48 (26.5%) | 74 (40.9%) | 80 (44.2%) | 52 (28.7%) |

## 6.Conclusion:

The Named-page Finding task of Terabyte Track requires to return the target document No.1 rank as possible. However, traditional information search methods with some other metrics, such as structural information and link analysis, can not achieve the requirements. As a result, a new re-ranking algorithm called PSM is proposed to improve the search performance, which uses a different way to treat the query and the document. Experiments showed that with the re-ranking algorithm, the results list can be effectively reordered, and the improvement of the main metric MRR is significant compared with the traditional information search methods.

Our future work may include combining the PSM algorithm with other traditional IR methods better. and also trying to find a way to make the formula in PSM algorithm embody more useful information.

## 7.References

[1] Robertson, S.E.and Walker, S.Okapi/Keenbow at TREC-8, In the Eighth Text Retrieval Conference(TREC 8), 1999, pp.151-162.

[2] Robertson, S.E.and Sparck Jones, K., Relevance weighting of search terms, Journal of the American Society of Information Science, Vol.27, No.May-June, 1976, pp.129-146.

[3] Ogilivie, P., Callan, J., Combining Structural Information and the Use of Priors in Mixed Named-Page and Homepage Finding, in Text Retrieval Conference 2003,Page 177