MICROSOFT RESEARCH ASIA AT THE WEB TRACK OF TREC 2003

Ji-Rong Wen, Ruihua Song, Deng Cai, Kaihua Zhu, Shipeng Yu, Shaozhi Ye and Wei-Ying Ma

Microsoft Research Asia
5f, Beijing Sigma Center
No.49, Zhichun Road Haidian District
Beijing 100080, P. R. China

INTRODUCTION

This is the first year that our group participates in the Web track of the TREC conference. Here we report our system and methods on the topic distillation task and the home/named page finding task.

All of our experiments are conducted on a Web search platform we designed and developed from scratch. We originally want to use an existing retrieval system such as Okapi or the full text search mechanism of SQL Server. But we soon find the limitations of such a strategy – these systems cannot fully support some important Web search functions such as link analysis and anchor text, and they also lack of the flexibility to arbitrarily adjust some parameters or add new ranking functions. So we decided to design and implement a research platform to let researchers to test various algorithms or new ideas easily and, also, to conduct the TREC experiments easily. We will introduce the framework of this system in the "Platform" section.

We feel that this year's topic distillation is more close to the real Web search scenarios. The target is to find a list of key resources for a particular (broad) topic and "key resources" are defined as the entry pages of websites. So, different from the previous years, we think that link analysis may play a positive role on identifying key resources in this year. As a consequence, we focus on using different link analysis techniques to enhance the relevance ranking. In particularly, we propose a novel block-based HITS algorithm to solve the noisy link and topic drifting problems of the classic HITS algorithm. The basic idea is to segment each Web page into multiple semantic blocks using a vision-based page segmentation algorithm we developed before. Then the main steps of the HITS algorithms, such as getting the seeds, expanding the neighbors using inlinks and outlinks, and calculating hub/authority values, can be performed at the block level instead of at the page level. Thus the noisy link and topic drifting problems can be effectively overcome. We will detail these techniques in the "Page Layout Analysis" and "Block-based HITS" section.

To our understanding, the biggest difference of this year's topic distillation task from last year is that, in general, only one most "suitable" page for each website should be returned as a top-ranked result. Any other page at the same website should not be included in the results or ranked highly since it is a "part of a larger site also principally devoted to the topic", despite that the page also "is principally devoted to the topic". Therefore, we construct a hierarchical site map for each website by building up the parent-children relationships of Web pages in the .GOV dataset. Then we apply a site compression technique to select the most suitable entry pages for websites among the retrieval results and return these entry pages as top-ranked

pages. This site compression method has proved to be quite effective to increase the p@10 precision if used appropriately, and will be introduced in the "Site Compression" section.

We totally submitted 5 runs for the topic distillation task and 3 runs for the home/named page finding task. In the "Experimental results" section, we will introduce these runs and their evaluation results.

PLATFORM

We built a Web search platform from scratch since we found traditional IR-like platforms cannot meet the requirements of comprehensive Web search algorithms. The architecture of the platform is shown at Figure 1.

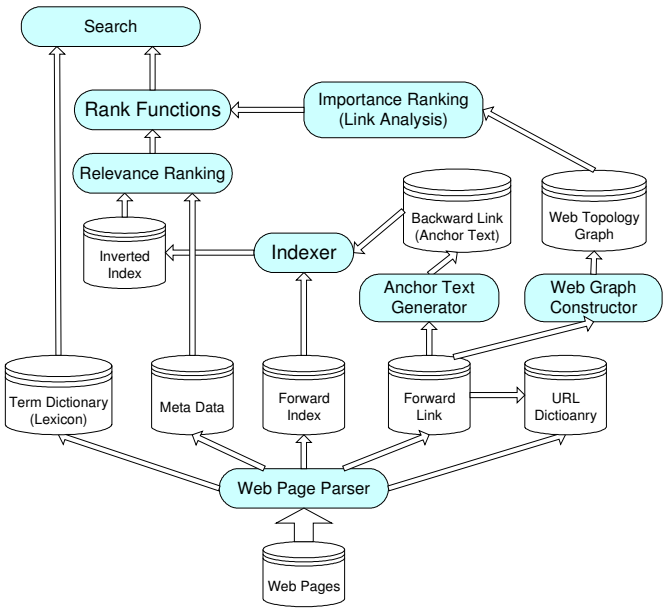

Figure 1. Architecture of our Web search platform

WEB PAGE PARSER

We built a robust Web page parser to extract the following information from each Web page:

1. Term hits – the type, format, position of each occurrence of each term in the page. Each term is assigned a unique term ID and a lexicon is built to store the term-ID mapping information during the parsing process. Similar to Google, we classify terms into five types:

   - ✧ Title: words in <title>…</title>
   - ✧ Meta: keywords extracted form <meta keyword="…">
   - ✧ URL: words that occur in hyperlinks
   - ✧ Anchor text: words in anchor texts from other pages and can only be obtained through a post-processing of all anchor texts

- ✧ Plain text: all the rest content words, and they are further divided into 6 categories:
  - H1_2: words in H1 and H2 tags
  - H3_6: words in H3 to H6 tags
  - STRONG: words with font type "bold", "italic", "underlined", etc.
  - Large: words with large font size
  - Medium: words with medium font size
  - Small: words with small font size

  All of the above information are extracted and put into a storage called "forward index".

2. URL and anchor text – all hyperlinks and their corresponding anchor texts are extracted and stored in a "forward link" storage. Also, each URL is assigned a unique ID and a URL dictionary is built. Two important functions rely on the forward link storage. One function is to add anchor texts to pages which the hyperlinks point to. Another is to construct Web graph and facilitate link analysis such as PageRank and HITS.

3. Meta data – some important meta data about each page, such as size, date and layout structure (will introduce later) are recorded in this storage.

INVERTED INDEX

There is a significant difference of the structure of our inverted index from general ones. Since we use an algorithm to segment web pages into semantic blocks, we add a block-level structure into the inverted index, as illustrated by Figure 2. Through this structure, we can identify each term hit at each block of each Web page. This structure is very critical to support our block-level search methods. Also, this structure occupies nearly the same (or less if we omit hits) storage space as page-level index and is compatible with page-level index.

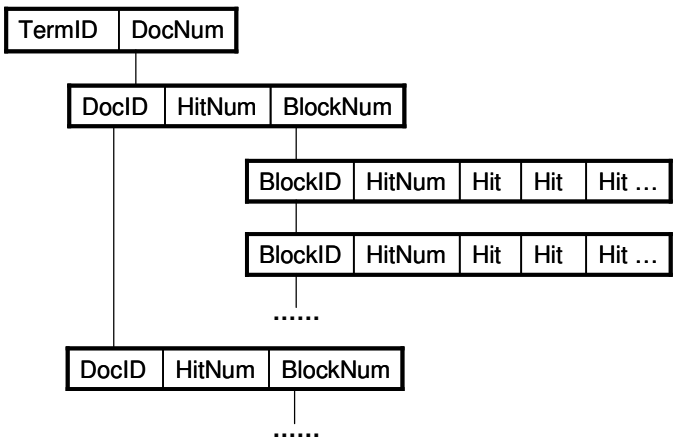

Figure 2. Structure of the inverted index

RANK FUNCTION

We use Okapi's BM2500 as our fundamental relevance ranking function. Considering the characters of Web search, we made some important modifications and augmentations to BM2500. First, it is allowed to set different weights to different term types and formats. For example, we can assign high weight to terms in titles of pages if we find title is more important for relevance ranking. Second, we use term proximity to adjust the relevance scores since it is observed that the distributions of query terms in a page significantly affect the relevance judgment.

PAGE LAYOUT ANALYSIS

As mentioned before, one distinguished feature of our platform is that each Web page is segmented into multiple semantic blocks. We think that a web page as a whole is not a good information unit for search because it often contains multiple topics and a lot of irrelevant information from navigation, decoration, and interaction parts of the page. We use a VIsion-based Page Segmentation (VIPS) algorithm to detect the semantic content structure in a web page based on visual cues such as color, line, font size, image, etc. For the sample page shown in Figure 3(a), the visual blocks are detected are shown in Figure 3(b) and the global layout structure is shown in Figure 3(c).

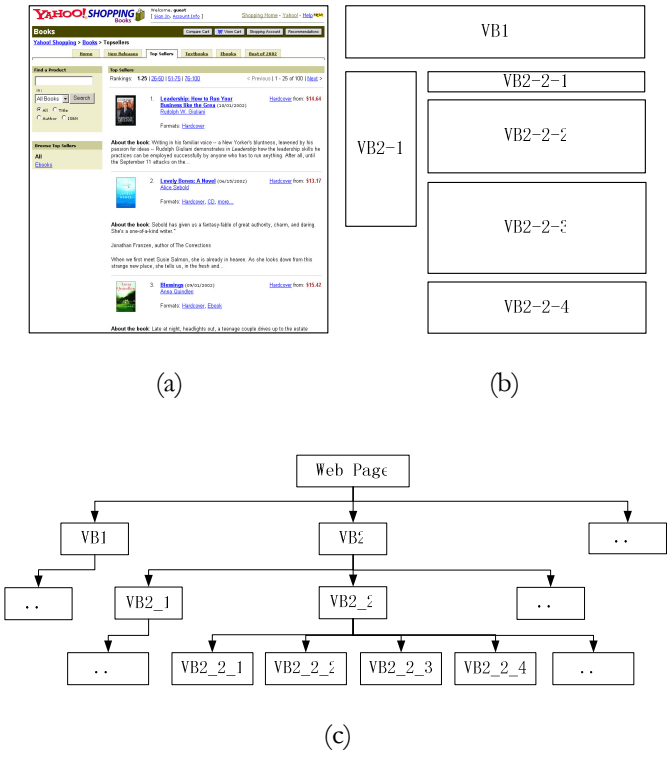

Figure 3. Vision-based layout structure for the sample page

VIPS can help to remove noisy information within a web page and detect multiple topics, and therefore it can be very beneficial to web information retrieval. In [8], VIPS is used in pseudo-relevance feedback to improve the quality of top ranked documents or blocks. About 27% improvement is achieved and it also wins over the DOM-based approach.

BLOCK-BASED HITS

Noisy link and topic drifting are two main problems in the classic HITS algorithm. Some links such as banners, navigation panels, and advertisements, can be viewed as "noise" with respect to the query topic. Generally, noisy links do not carry human editorial endorsement, which is a basic assumption in topic distillation. Also, hubs may be "mixed", which means that only a portion of the hub content may be relevant to the query. Most link analysis algorithms treat each Web page as an atomic, indivisible unit with no internal structure. This leads to false reinforcements of hub/authority calculation.

By segmenting a Web page into separate semantic blocks, we implement a modified HITS algorithm at block level. We hope to verify that page segmentation is an effective way to overcome the noisy link and topic drifting problems of HITS, to some extent. Below are the main steps of the block-based HITS algorithm.

Step 1: A *start set* of pages matching the query is fetched by a block retrieval algorithm (described below), and the top 200 pages are used.

Step 2: The start set is augmented by its neighbors, with no limitation on inlinks. When the neighborhood graph is expanded using outlinks, only those outlinks in blocks with high block ranks are used to expand the *neighborhood set.*

Step 3: Prune the neighborhood blocks by a query expansion algorithm (also described below). First, we use the query expansion algorithm to get a list of blocks with relevance scores. Then these blocks are intersected with the blocks in the *neighborhood set.* The blocks in the intersection set are labeled using the scores of the query expansion step. Finally, we use the median value as a threshold to prune the nodes whose weights are below this threshold. The resulting set is used as a new neighborhood set.

Step 4: In the neighborhood set, if there are k edges from pages on a first website to a single page on a second website, we assign each edge an authority value of $1/k$. This weight is used when computing the authority score of the page on the second website. If there are t edges from a single page on a first website to a set of pages on a second website, we assign each edge a hub weight value of $1/t$. Finally we remove isolated nodes from the graph.

Step 5: Only each root block (i.e. the whole page) can be assigned an authority score. Every leaf block has its hub score. So we build up a $m \times n$ matrix in which m stands for the number of blocks in the neighborhood set and n stands for the number of pages in the neighborhood set.

Generally, the calculating results of our block-based HITS algorithm have a very "sharp" distribution – generally only the first 15~30 authority or hub values are greater than zero. And all of the remaining pages have a zero value. Therefore, only those most densely linked collection of hubs and authorities in the neighborhood graph can be detected and distinguished by the algorithm. But this is not a serious problem for us since P@10 is used as the evaluation measure this year. So such a value distribution is sufficient to provide the differentiated capability for the top 10 results.

BLOCK RETRIEVAL

Similar to passage retrieval, block retrieval performs the retrieval task on the block level and aims to adjust the rank of documents with the blocks they contain. The block retrieval algorithm contains the following steps:

Step 1. Initial Retrieval: An initial list of ranked web pages is obtained by using the general page level retrieval method and a page-level rank called *PR* is obtained.

Step 2. Page Segmentation: In this step, the page segmentation algorithm VIPS is applied to partition all of the retrieved pages into blocks. All the extracted blocks form a block set.

Step 3. Block Retrieval: This step is similar to Step 1 except that pages are replaced by blocks. The same queries are used to produce a block-level rank called *BR* for each block.

For each page, the block with the highest *BR* rank is selected and its rank is called *BRMax*. In our experiments, a combination of *PR* and *BRMax*, formatted as $\alpha \cdot rank_{PR}(d) + \beta \cdot rank_{PR\_BRMax}(d)$, is used as the final rank of each page.

QUERY EXPANSION

In our experiments, we use pseudo-relevance feedback as a basic query expansion method. Its basic idea is to extract expansion terms from the top-ranked pages to formulate a new query for a second round retrieval. The effect of query expansion method strongly relies on the quality of selected expansion terms. Since our VIPS algorithm can group semantically related content into a single block, the term correlations within a block will be much higher than those within a whole page. With the improved term correlations, high-quality expansion terms can be extracted from blocks and then used to improve retrieval performance. The query expansion algorithm contains the following steps:

Step 1 – Step 3 are the same as those of block retrieval. We get a page rank *PR* for each page and a block rank *BR* for each block after these steps.

Step 4. Expansion Term Selection: Top blocks are used for expansion term selection. We use an approach similar to the traditional pseudo-relevance feedback algorithm to select expansion terms. All terms except the original query terms in the selected blocks are weighted according to the following term selection value *TSV*:

$$TSV = w^{(1)} * r / R$$

where $w^{(1)}$ is Robertson/Sparck Jones weight [7]. *R* is the number of selected blocks, and *r* is the number of blocks which contain this term. In our experiments, top 10 terms are selected to expand the original query.

Step 5. Final Retrieval: The weights for the expanded query terms are set as the following:

- For original query terms, new weight is $qtf \cdot 2$ where *qtf* is its term frequency in the query;
- For expansion query terms, new weight is $1 - (n-1)/m$ if the current term ranks nth in *TSV* rank. *m* is the number of expansion terms and is set to 10 in our experiments.

Then the expanded query is used to retrieve the data set again for the final results.

SITE COMPRESSION

It is a normal phenomenon that multiple Web pages from the same sites are ranked highly for a given query. This is not a problem if the objective is to find relevant pages. But this year's topic distillation task targets to find a list of key resources for a particular topic. Key resources are defined as the entry pages for websites and other pages should be discarded. So we should try to find as many different websites (represented by their entry pages) as possible within the first ten results. So finding a method to detect the entry page for each website is very important.

We construct a hierarchical site map for each website by building up the parent-children relationships of Web pages in the .GOV dataset. Then we apply a site compression technique to select the most suitable entry pages for websites at the retrieval results and return these entry pages as top-ranked pages. Figure 4 is a sample site map of the "fitness.gov" website.

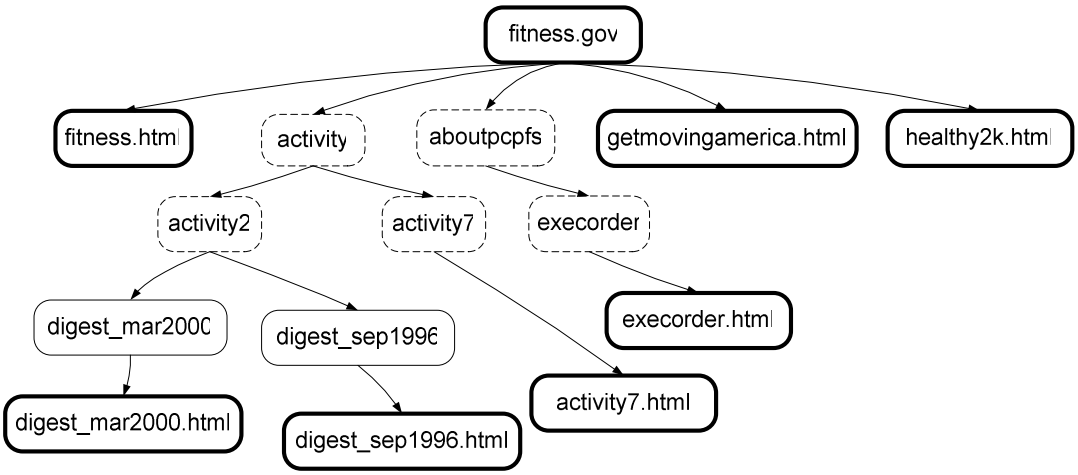

Figure 4. Site map

The solid boxes represent a page appearing in the search results and the dashed boxes represent virtual directories which do not appear in the results. Take the topic of "physical fitness" as an example, below are the pages and their ranks from the "fitness.gov" website among the top 1000 results.

2. http://fitness.gov/activity/activity2/digest_mar2000/digest_mar2000.html

3. http://fitness.gov/aboutpcpfs/execorder/execorder.html

16. http://fitness.gov/

21. http://fitness.gov/getmovingamerica.html

23. http://fitness.gov/healthy2k.html

……

To judge if a parent page (solid box) can represent all of its children results, we check the following two conditions:

- If the parent page has more than three child pages (with solid box) in the top 1000 results

- If the parent page has more solid children than dashed children

If any of the two conditions is met, the children can be represented by their parent page, and the rank of the parent page is assigned with the maximum of its rank and its children's ranks. The above rules are applied to each site maps in the results recursively from bottom up. This site compression method proved quite effective in terms of increasing P@10 if used appropriately.

EXPERIMENTS

We totally submitted 5 runs for the topic distillation task and 3 runs for the home/named page finding task. Below is the list of the runs:

- MSRA1001 – The augmented BM2500 (by adding term weights and term proximity) is used as the rank function. Site compression is used to post-process the ranking results and only one page from each website is kept in the top 10 results.

- MSRA1002 – Similar with MSRA1001 only except that the site compression step allows at most 2 pages from each website are kept in the top 10 results. Notice that we do not use any link analysis technique in MSRA1001 and MSRA1002. We want to use these two runs as our baseline to compare them with the runs with link analysis.

- MSRA3 – MSRA1001 is first used to get a result list. Then the importance value calculated by PageRank is used to re-rank the results.

- MSRA4002 – MSRA1002 is first used to get a result list. Then the authority value calculated by our block-based HITS algorithm is used to re-rank the results. Notice that the site compression step in MSRA1002 is executed after the HITS step.

- MSRA4003 – MSRA1001 is first used to get a result list. Then the authority value calculated by our block-based HITS algorithm is used to re-rank the results. Also, the site compression step in MSRA1001 is executed after the HITS step.

  The above 5 runs are related to the topic distillation task.

- MSRANP1-3 – These are the 3 runs related to home/named page finding task. The rank function is similar with MSRA 1001, except that no site compression is used and term weight settings are different.

TOPIC DISTILLATION

Table 1 shows results of our five topic distillation runs and the techniques used in each run are listed in Table 2.

We found that term weight settings are important for relevance ranking. We use a greedy algorithm to automatically learn weights for different term types and format by using the data of TREC'02. In our experiments, PageRank do not show significant improvement on P@10. But we found that the pages returned are perceived well when browsing. Therefore, we have no conclusion on whether PageRank is useful based on

the experiments. Block-based HITS shows a steady improvement on retrieval performance. The two runs containing block-based HITS, MSRA4002 and MSRA4003, got the best two p@10 and average precision. We also find that the performance of block-based HITS is significantly better that PageRank at the .GOV dataset, which further affirms that PageRank is not suitable to be used in a relatively small or moderate dataset.

Run	Precision@10	Average precision	R-Precision
MSRA1001	0.0960	0.0699	0.1027
MSRA1002	0.1100	0.0824	0.1078
MSRA3	0.1040	0.0933	0.1016
MSRA4002	0.1160	0.1027	0.1354
MSRA4003	0.1140	0.0946	0.1052

Table 1: Topic distillation results of 5 runs

Run	Term Weight	Term Proximity	PageRank	Block-based HITS	Site Compression*
MSRA1001	Y	Y			Y (1)
MSRA1002	Y	Y			Y (2)
MSRA3	Y	Y	Y		Y (2)
MSRA4002	Y	Y		Y	Y (2)
MSRA4003	Y	Y		Y	Y (1)

Table 2: Techniques used in topic distillation runs

* Y(1) means that a run uses site compression and keeps only 1 page from a site in the top 10 results, while Y(2) means that a run uses site compression and keeps at most 2 pages from a site in the top 10 results.

Site compression is proved to be quit effective in terms of increasing P@10 (Figure 3). However, to retain only one page for each website may be risky since the site compression method cannot identify the entry pages with 100% accuracy. So we can see that MSRA1001 only achieve a slight increase of P@10 in comparison with the baseline (without site compression). But if we adapt a conservative strategy to retain at most two pages for each site, a significant increase of P@10 of MSRA1002 is shown.

Run	Precision@10	Average precision	R-Precision
Baseline (without site compression)	0.0940	0.0966	0.0995
MSRA1001	0.0960	0.0699	0.1027
MSRA1002	0.1100	0.0824	0.1078

Table 3: Effect of site compression

Finally, run MSRA4002, which combines augmented BM2500, block-based HITS and conservative site compression, achieve the best performance on P@10, average precision and R-Precision.

NAMED PAGE FINDING

Run	Average Reciprocal Precision	Named pages in top 10	Named pages not found	Features
MSRANP1	0.651	253 (84.3%)	27 (9.0%)	Anchor_weight = 3 Title_weight = 1.1 Other weights = 1
MSRANP2	0.540	214 (71.3%)	56 (18.7%)	Linear combined with term proximity
MSRANP3	0.556	218 (72.7%)	51 (17.0%)	Anchor, title and url only

Table 4: Named page finding results and features

For the named page finding task, we mainly focus on setting proper weights for different term types and the weights are learned based on the data of TREC'02 named page finding task. We discovered that anchor

plays the most important role in this task as its weight is highest, and title is the second one. Run MSRANP1 is the baseline with term weight settings as shown in Table 4. MSRANP2 is a run by combining MSRANP1 and term proximity linearly. But its average reciprocal precision decreases quite a few. In run MSRANP3, only anchor, title and url are used in the rank function and the result indicates that it lost 8% more named pages than the run uses all fields.

REFERENCES

1. Bharat, K. and Henzinger, M., Improved Algorithms for Topic Distillation in a Hyperlinked Environment, Proceedings of 21st ACM International Conference on Research and Development in Information Retrieval, 1998
2. Brin, S. and Page, L., The Anatomy of a Large-Scale Hypertextual Web Search Engine, In the Seventh International World Wide Web Conference, Brisbane,Australia, 1998.
3. Callan, J. P., Passage-Level Evidence in Document Retrieval, In Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, 1994, pp. 302-310.
4. Chakrabarti, S., Joshi, M., and Tawde, V., Enhanced topic distillation using text, markup tags, and hyperlinks, In Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval , ACM Press, 2001, pp. 208-216.
5. Kleinberg, J., Authoritative sources in a hyperlinked environment, In Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, 1998, pp. 668-677.
6. Robertson, S. E. and Walker, S., Okapi/Keenbow at TREC-8, In The Eighth Text REtrieval Conference (TREC 8), 1999, pp. 151-162.
7. 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.
8. Yu, S., Cai, D., Wen, J.-R., and Ma, W.-Y., Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation, In Proceedings of the Twelfth International World Wide Web Conference (WWW2003), May 2003.