PARADISE Based Search Engine at TREC 2009 Web Track

Dongdong Shan, Dongsheng Zhao, Jing He, and Hongfei Yan
School of Electronic Engineering and Computer Science
Peking University, Beijing, China, 100871
{sdd, zds, hj, yhf}@net.pku.edu.cn

ABSTRACT

In this paper, we introduce the PARADISE search engine in TREC09 Web track. PARADISE is the abbreviation for Platform for Applying, Research and Developing Intelligent Search Engine, which is a search engine platform developed by SEWM group, Peking University. The system is designed to support both English and Chinese information retrieval. This system preprocessed and indexed the five hundred million web pages for this year's Web Track. In the preprocessing stage, the templates were removed, the encoding were identified and unified, and the anchor texts and InLink information are extracted with the mapreduce framework (using Hadoop in this system). In retrieval, our runs used an extension of BM25. This model distinguishes terms from different fields and integrated both term counts and position information. Furthermore, some web based features are also considered.

Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Systems Issues, Retrieval Models

General Terms

Performance, Design, Experimentation.

Keywords

System Design, Information Retrieval, Term Proximity.

1. INTRODUCTION

PARADISE[1] was developed by Search Engine and Web Mining group in Peking University. It aims to provide a search engine platform for processing large volume of data. The Web track requires retrieving documents on a very large document collection, so it becomes a challenge for both effectiveness and efficiency.

PARADISE system contains several important components: store, preprocessing, indexing and retrieval. We would introduce both the architecture and algorithms in detail. Furthermore, we would analyze our performance in Web track this year.

2. Task

This year's Web Track uses a new collection named ClueWeb09. It contains one billion web pages in different languages. The total size of this new collection is 25TB, and the compressed size is 5TB, which is significantly larger than earlier collections. This year's task focuses on the English corpus in the test collection，which involves a total of 504 million English web pages with a size of 1.94TB after compression.

In this task we used 24 machines. 23 machines are used to process the data, such as content extraction, data indexing, information retrieval. The rest is used as a front-end service machine. It interacts with the 23 machines when they run queries. It submits queries and collects the results returned by those machines. Each of the machines has two 2.8 GHz Intel Pentium CPUs, 4GB memory, and 1.6TB of local SATA disk.

In addition, we have a cluster of 30-machine running Hadoop distributed computing environment. We use it to merge anchor texts, compute PageRank, and find duplicate pages.

3. Document Representation

In our system, we have two kinds of documents. The first kind of document is a normal document, such as HTML, PDF, and MP3. The second one is the document that has been preprocessed by preprocessing system. This kind of document has a certain format. The index system only reads this kind of document. We will take HTML page as an example to introduce the second kind of document contains.

As we know, a HTML page contains much information, such as URL, title, body, description and anchor texts. There is also some specific information, such as importance and duplicate degree of this document. The preprocessing system is mainly used to extract these information from the original web page, and then save those information into store system. Therefore the indexing system can read the formatted document for store module.

In this year's task, we use the following information as contents of the document: title, body, anchor text, URL's text, and InLink value. The title part contains the text in the

<title> tag. We don't extract the real title of this document. Body mainly contains HTML text without HTML tags. Anchor text includes text information showing that other pages point to this page. URL text contains text information belongs to the URL in this page. Inlink degree represents the total number page links pointing to this page.

4. Data Store Module

Web pages are allocated to different data servers by the hash values of their URLs. The hash function is designed in a way that pages from the same host are hashed to the same data server. In the allocating process, each page is assigned a unique 64-bit ID, the first 24 bits of which represent the data server and the last 40 bits is the page's sequence number on that data server.

A data server partitions the received web page data into a set of files, whose size can be configured. In this task, it is configured to 1GB. Data server's application interfaces provide sequential scan and random access functions. In this task, each data server maintains about 130GB web page data and processes 2,200,000 pages on average.

Two data formats are used on data servers. In one of them, namely the Tianwang format, web pages are simply sequentially stored, which provides good sequential scan performance but it is slow when random accessing. However, the whole processing rarely performs random access to the pages, so this format works well. Alternatively, the other format supports quick random access but not quick sequential scan. This format is designed for document summary and snapshot services in which case page data need be fetched efficiently.

Information after preprocessing includes the original web page and the result of preprocessor. We also store that information, such as the titles and the main contents on those store servers.

5. Data Preprocessing and Indexing

The main target of preprocessing system is to extract useful information from various files, such as HTML, PDF, and MP3 etc. for retrieval system. When it extracts information, it usually contains the following steps: web page encoding transformation, page normalization, noise elimination, anchor text extraction, link-analysis, and duplicate pages elimination etc.

Web page encoding transformation translates various coding of html pages into one certain coding. Page normalization repairs the missing tags in HTML page. Noise elimination removes useless information like advertisements from the web page. We extract anchor text and URL pairs on multiple machines when eliminating noise information in HTML pages.

Duplicate pages elimination aims to find the duplicate pages, and to remove the Duplicate pages from the collection. It is useful in web search. But when we did experiments with GOV2 dataset, we found that lots of pages were removed by our program so that the recall was very low. In this task, we do not use this step when preprocess the collection. We use a simple version to remove the noise in web pages, by just removing the tags of HTML page.

We use Hadoop to merge anchor texts and compute the InLink degree for each page. The computing data size is about 1.7 TB. We have two copies for replication in Hadoop. The Map function [2] reads huge <URL, anchor text> key-value pairs from link list file. We found there are large amounts of duplication in anchor texts. So we encode the anchor texts and compress the data by merging the strings of anchor texts. This reduced 80%-90% storage compared with raw data. The reason why we use Inlink instead of PageRank is that computing InLink is more efficient.

The indexing system reads information about each web page from store module. In this year's task, we index four kinds of texts: title, body content, URL text, and anchor text. We also store InLink values, page ID and URL in index file. Indexing system tokenizes the text, removes stop words and stems word with poster algorithm. Then it builds the inverted file. It also writes the special information such as InLink value and the length of texts into a special file. This kind of files should be read fast, so this information always is kept in memory.

We submit three official runs for evaluation. They are generated by different information. Run1 and Run2 use the same copy of index, which only contains title and body. The index used in Run3 contains all information as we mentioned before. We also reserve the stop words in index using by Run3. Although it makes the index much bigger, stop words can help retrieval under some condition.

6. Retrieval Model

The retrieval model is based on BM25. We slightly extend this formula. Every field is extended according to its weight during the retrieval process [3]. InLink, term proximity and term occurrence are also taken into consideration. The final score formula of a document is

$$Score = c \times (\alpha \times Score_{BM25} + \beta \times Score_{InLink} + \gamma \times Score_{TP}) \quad (1)$$

where β is set to 0 in Run1 and Run2. Table 1 shows the three official runs used information. Parameter c is the term occurrence score, which is calculated as

$$c = \frac{c_1 + q_{occ}}{c_2 + q_{total}} \quad (2)$$

In this formula, q_{total} is the number of unique terms in the query, and q_{occ} represents number of terms which have occurrence in the current document. c_1 and c_2 are two smoothing parameters. We find it can get the best results when $c_1 = 6$ and $c_2 = 5.5$ in GOV2 data collection. $Score_{BM25}$ is the score computed by BM25 model;

$Score_{InLink}$ represents contribution made by InLink; and $Score_{TP}$ is defined as the score of term proximity. Each of the three is described below.

$Score_{BM25}$ is calculated as normal BM25 formula with a small modification, that is, the term frequencies is extended in each field. For example, a term t appears once in title, 3 times in content. If we define the title weight as 5 and content weight as 1, the final term frequencies is 1*5+3*1 = 8. The BM25 parameters are set

Title and content are treated equally inRun1. The computation of $Score_{tp}$ and the value of γ are different between Run1 and Run2. In Run2 and Run3, the proportion of title and content is set to 2:1. $Score_{InLink}$ is calculated [4,5] as

$$Score_{InLink} = \frac{\ln N}{\ln N + c_3} \quad (3)$$

where N is the value of InLink, and c_3 is a parameter. The parameter β is set to 0 in Run1 and Run2. We set c_3 is 0.5 in our runs.

$Score_{TP}$ is described as follow. For a query $t_1t_2\ldots t_n$ with n terms, there exists n-1 adjacent term pairs: t_1t_2, $t_2t_3\ldots t_{n-1}t_n$. For each adjacent term pair t_it_{i+1} in different fields, we can compute the minimum distance between t_i and t_{i+1}, which is denoted as $MinDist_{i,f.}$ The score is calculated [6,7] as

$$AdjScore_{t_it_{i+1}} = \sum_{f\in F} \frac{Boost_f}{MinDist^2_{i,f}} \quad (4)$$

where $Boost_f$ denotes the weight of field f, F is the set of indexed fields. The score of all adjacent t_it_{i+1} in all fields is

$$RawScore_{TP} = \sum_{i=1,2..n-1} AdjScore_{t_it_{i+1}} \quad (5)$$

The final $Score_{TP}$ is

$$Score_{TP} = \frac{RawScore_{TP}}{RawScore_{TP} + c_4} \quad .(6)$$

C_4 is a smoothing parameter, which is set to 0.5 in our experiment. We use RawScoretp in Run1, and Scoretp in Run2 and Run3.

Table 1 The information and method each run used

Run	Anchor Text	URL Text	InLink	Term proximity
1 pkuTp	N	N	N	Y
2 pkuStruct	N	N	N	Y
3 pkuLink	Y	Y	Y	Y

7. Result and Analysis

In this section, we analyze of three official runs we submitted. Though, the results are not as good as we expected, we find some important and worth-exploring factors.

In this year's task, the average length of 50 topics is 2.1, which is shorter than that of previous Terabyte Track by 1. In addition, there are also many single-word topics. We turn our system parameters based on GOV2 dataset using topic 701~850 without any special optimization on short queries. We also remove the stop words from the topics, which make the topics shorter. The description of topics is show in table 2.

Table 2. Length of topics in Wt09 and GOV2

Topic Length	Wt09(total 50)	701-850(total 150)
1	17(34%)	2 (1.3%)
2	17(34%)	39 (26%)
>2	16(32%)	109(72.7%)

Due to these short topics, we give an analysis of how they affect the performance of our system. Among 17 one-word topics, 14 topics need to find the homepage, only three topics are purely information topic (wt09-04: totlet, wt09-12: djs, wt24: diversity). In official Run1 and Run2, these methods only find one topic which has relevant results in top10. For other 16 topics, we don't find any relevant document in top10. Link method (Run3) can find at least 7 topics which at least one relevant document in top 10. From the above results, we can find link method is better than previous two methods we mentioned for one-word topics. It may because link method combines the URL content and link information. However, it performs worse than the previous two methods for information topic wt09-12.

Table 3 The number of topics find at least one relevant document @5,@10

Topic Length	Run1	Run2	Run3	Med
Rel Retr @ 5(Len=1)	1	1	5	4
Rel Retr @ 10(Len=1)	1	1	7	7
Rel Retr @ 5(Len=2)	5	5	5	7
Rel Retr @ 10(Len=2)	6	5	8	12
Rel Retr @ 5(Len>2)	8	8	7	10
Rel Retr @ 10(Len>2)	10	10	9	11
Rel Retr @ 5(all)	14	14	17	21
Rel Retr @ 10(all)	16	16	24	30

For two-word queries, our runs give bad results due to several reasons. First, 13 of these 17 topics are needed to find homepage. Second, we filter out stop words and stems word which makes the topics shorter. However, we need further experiments to find out the true reasons. We also find that the methods using document structural information performs slightly worse than that without structural information. Maybe we use different term proximity scores. Most of these two-word topics are phrases, while our system doesn't optimize for them. Although we use the proximity techniques, we limited the maximum proximity score is 1. For other 16 longer queries, we find more relevant documents in top 5 ranks and top 10 ranks. This may suggest that our system is more robust for longer queries.

Since our system finds none relevant docs in top 10 for nearly half of all topics, the whole score we get is much lower than that of GOV2.

Table 4. Effectiveness of office run

Run	bpref	map	P5	P10
1.pkuTp	0.170	0.059	0.132	0.148
2.pkuSturct	0.171	0.062	0.132	0.146
3.pkuLink	0.159	0.045	0.108	0.116

8. Conclusion

The Web track provides a good test bed for retrieving documents from a very large Web document collection. We find out some deficiency of the PARADISE system, including: 1) It performs worse for short queries; 2) There is no query analysis system. It is supposed to perform better if we can distinguish multiple query intents; 3) It has not employed much information from web structure, such as PageRank.

9. Acknowledgments

This work is partially supported by China MOE under the grant number 2008107, NSFC under the grant number 70903008 and 60933004, and 863 Grant 2007AA01Z154. We also thanks to Kai Fan who help us to calculate InLink values and merge anchor texts via Hadoop Cluster.

10. References

[1] "http://sewm.pku.edu.cn/src/paradise/."

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters." OSDI 2004

[3] S. Robertson, H. Zaragoza, and M. Taylor, "Simple BM25 Extension to Multiple Weighted Fields," CIKM 2004, pp. 42-49.

[4] T. Upstill, N. Craswell, and D. Hawking, "Query-independent evidence in home page finding," *ACM Transactions on Information Systems (TOIS)*, vol. 21, 2003, pp. 286-313.

[5] N. Craswell, S. Robertson, H. Zaragoza, and M. Taylor, "Relevance weighting for query independent evidence," ACM, 2005, p. 423.

[6] C.L. Clarke and B. Lushman, "Term Proximity Scoring for Ad-Hoc Retrieval on Very Large Text Collections," *Third Text*.

[7] R. Song, M.J. Taylor, J. Wen, H. Hon, Y. Yu, and M.R. Asia, "Viewing Term Proximity from a Different Perspective," *Work*, 2008, pp. 346-357.