OASIS at NTCIR-5: WEB Navigational Retrieval Subtask

Vitaly KLYUEV
Software Engineering Lab
The University of Aizu
Tsuruga, Ikki-machi, Aizu-Wakamatsu city, Fukushima, 965-8580, Japan
vkluev@u-aizu.ac.jp

Abstract

We experienced negative results participating in the NTCIR-5 Web Navigational Retrieval Subtask: the OASIS system, which is a distributed search system based on VSM and full text indexing, failed to retrieve relevant documents from the huge data set of Japanese Web pages when the number of relevant documents in the collection was relatively small. This paper describes our approach and the techniques used.
Keywords: *search engine, distributed system, full-text phrasal indexing, vector space model.*

1 Introduction

At the 5^{th} NTCIR Workshop, OASIS participated in the NTCIR-5 Web Navigational Retrieval Subtask (the Navi-2 Subtask). OASIS stands for the Open Architecture Server for Information Search and Delivery [4]. Our aim was to test the ability of the system to search huge datasets of Japanese documents.

The basic idea of the OASIS approach is the following. The OASIS service presents a distributed system of search engines in the Internet. The standard server usually creates and supports one or more subject-specific (topic related) indices. The user query is only propagated to a subset, which possibly contains the requested information. After merging, returned results are presented to the user in the usual search engine manner [6].

To meet all requirements of the Navi-2 Subtask, the core of the system was changed. The Vector Space Model behind the system and an idea to use the distributed approach to index the data remained.

A test dataset provided by the organizers was divided into three parts without any topical principle and distributed between three different servers. As a result, the propagation and merging mechanisms were changed: Every query was propagated to all sets of servers. An approach adapted to merge retrieval results is discussed in section 2.

We applied a classical variant of the Vector Space Model (VSM) to represent documents and queries in the memory of a computer. They were considered as vectors in the term space. A measure of similarity between the two vectors was computed. To calculate the value of the jth entry in the vector corresponding to document D_i, the following formula was used [2]:

$$d_{ij} = tf_{ij} * idf_j$$

where tf_{ij} is the number of occurrences of term t_j in document D_i, $idf_j = \log(\frac{d}{df_j})$. d is the total number of documents in the data set and df_j is the number of documents which contain t_j. A similarity between query Q and document D_i is defined by the product of two vectors in VSM. To compute its value, we applied the standard formula as follows:

$$S(Q, D_i) = \sum_{j=1}^{t}(w_{qj} * d_{ij})$$

where w_{qj} and d_{ij} are term weights of vectors Q and D_i respectively; t is the total number of terms.

The full-text indexing technique was applied to the dataset offered by the organizers.

2 Methods Used

We applied the techniques characterized below to enhance the vector space model implemented:

1. The indexing units used by our system are words extracted by the Mecab lexical analyzer and pseudo-collocations generated automatically. We utilized the following simple approach to compute them. Every term generated by Mecab has a part of speech mark. Our algorithm employed noun, adjective and unknown word marks. Each pseudo-collocation, consisting of two or three terms, was constructed from two to three nouns or unknown words located next to one another in a document. Two term collocations were built from an adjective and a noun or an adjective and

an unknown word. If their position in the document was next to each other, a new collocation was taken into account.

Japanese stop words from the list provided in [1] were applied to eliminate common terms.

2. We took into consideration how often each term appears in document headings (in the title section) and enhanced weights (1.5 times) accordingly.

3. Weights of the terms gathered around URLs were increased in a 'wave' style manner. The following formula was applied:

$$w_{i,u}^{w} = w_{i,u} + \frac{1}{2 * |i|} * w_{i,u} for i = \pm 1, \ldots, \pm 5,$$

where i as a distance in words from the URL u; $w_{i,u}$ is a weight of the term.

4. A two-stage retrieval technique was used to get and merge results obtained from the servers: First, terms from the NARR element of the corresponding topic description were submitted to each server. The aim was to get documents including at least one searched term. Second, all documents after retrieval were put into one pool. An examination of this pool gave the final result of the search. The latter query included terms from the TITLE element of the topic description.

5. Alternatively, the Borda-fuse algorithm introduced in [3] was adapted to the configuration of our system to merge the search results and assign the final relevant score. As we mentioned in section 1, there was no topical distribution of data between servers: They served different parts of the dataset. Each of our three servers retrieved up to 30 documents in response to a query submitted. The top ten documents from each returned list were given a new score according the formula.

$$S(D_i^k) = \frac{101 - i}{N_k}, i = 1, \ldots, 10$$

where D_i^k is ith ranked document retrieved by server $\hat{k}$; N_k is a number of documents retrieved by server $\hat{k}$. Because each server kept different sized data, the damper coefficients were utilized to adjust the obtained score. These coefficients reflect the size of the data of each server: They are 0.5, 0.3, and 0.2.The rest of documents from the retrieved lists kept their initial score assigned by the corresponding server. Finally, all retrieved documents were ranked in order of the points calculated. This scoring mechanism requests the servers to get only a small number of documents because fewer hits returned by a server, the higher the score given to them, and the higher the place on the list presented to the end user they receive.

Table 1. Parameters of the Servers

Number	Processor	Memory	Hard Drive
1	Dual Intel Xeon 2.8GHz	2GB	500GB
2	Intel 1.7GHz	1GB	1TB
3	Intel 1.8GHz	2GB	300GB

3 System Description

The prototype which was used in the Navi-2 Subtask included three servers on the base of PC compatible computers running Linux 9.0 and Linux 7.3. Their parameters are presented in Table 1.

Servers were attached to each other using a gigabit network connection. Full-text indexing took approximately 2 months. Half of the data set was put on server 2. The second half was distributed between server 1 and 3. Server 1 kept a slightly larger portion of data.

4 Search Results

Only mandatory queries were submitted to the system: They were taken from topics $1001 - 1399$. A retrieval process was carried out in a fully automatic way. The results of three official runs were submitted as an outcome of our tests. They are OASIS-01, OASIS-02, and OASIS-03. The search techniques explained in section 2 (items 1 to 4) were applied to run OASIS-02 and other sets of techniques (items 1, 2, 3 and 5) were activated for run OASIS-03. During the OASIS-01 run, one of our servers generated an error and stopped. This run did not make a retrieval from the whole data set. Affected topics were $1001 - 1146$.

The system failed in the retrieval of relevant documents. Table 2 gives some examples. It introduces the number of retrieved documents by OASIS (OASIS-03 run) in response to queries submitted and the number of relevant documents (A and B categories: relevant and partially relevant) defined by assessors.

5 Failure Analysis

The problems we faced in conducting this research were as follows:

- The amount of data was critical for the hardware we used: a deficit of the hard drive space. A portion of data (about $200GB$) was not indexed. We were worried about overrunning indexes and allocating working space big enough on hard disks.

Table 2. Retrieval Statistics: Some Examples

Query number	Number of retrieved documents	Number of relevant documents in the collection
1003	5	2
1004	40	3
1005	22	1
1006	14	12
1008	3	5
1010	10	12
1394	21	6
1395	40	21
1397	37	1
1398	38	1

- Indexing and searching such a dataset were testing the system itself and checking its ability to manage large amounts of data: Some bugs were found and fixed.

- We dealt with a deficit of time to test different approaches.

- Indexing *raw* data was not successful for our system because of a large amount of errors on Web pages in the corresponding data set. In connection with this issue the following note is still important: Any parser which is designed to run on the entire Web must be capable of handling a huge array of possible errors. These include typos in the HTML tags, kilobytes of zeros in the middle of tags, HTML tags nested hundreds deep, etc [5]. The *cooked* data prepared by the organizers did not include the HTML information. We did not take into account the HTML structure of the data. The *title* tag was only an exception.

- Link information was not taken into account because of problems with their application.

We failed to retrieve relevant documents applying an improved VSM approach to the huge dataset when the number of relevant documents in the collection was relatively small. The results should be investigated carefully. It was not a problem of the merging technique we used: A merging process did not discard relevant pages. Our servers did not retrieve relevant documents. We see the following reasons for this:

- Some bugs remain in our software.

- Probably, adding heuristics and incorporating negligible improvements to the commonly used models (VSM, probabilistic, etc.) cannot be an efficient solution to discover a small amount relevant documents and retrieve them successfully from very huge data collections. We think they cannot work well for this task.

- Scientists working in the text information retrieval area need a new language model applicable to natural languages (Japanese, English, etc.) to design new systems which can be tools for the Navi-2 Subtask.

References

[1] Mnogosearch web search engine software (a list of japanese stopwords). http://www.mnogosearch.org/.

[2] David A. Grossman and Ophir Frieder. *Information Retrieval: Algorithms and Heuristics*. Kluwer Academic Publishers, 2000. (ISBN: 0-7923-8271-4).

[3] J.S. Asiam and M. Montague. Models for metasearch. In *SIGIR 2001*, pages 276–284, New Orleans, Luisiana, USA, 2001.

[4] V. Kluev. Compiling document collections from the interner. *SIGIR Forum*, 34(2):9–14, Fall 2000.

[5] Sergey Brin and Lawrence Page. The anatomy of a large–scale hypertexual web search engine. In *Proceedings of 7th International World Wide Web Conference (WWW-7)*, 1998.

[6] V. Kluev. Web search experiments using oasis. In *Proceedings of the Third NTCIR Workshop Meeting*, Tokyo, Japan, 2002.